Range Commanders Council
DATA REDUCTION AND COMPUTER GROUP

STANDARD PROTOCOL FOR REAL-TIME COMPUTER TO COMPUTER DATA EXCHANGE

WHITE SANDS MISSILE RANGE
KWAJALEIN MISSILE RANGE
YUMA PROVING GROUND
DUGWAY PROVING GROUND
ABERDEEN TEST CENTER
NATIONAL TRAINING CENTER

ATLANTIC FLEET WEAPONS TRAINING FACILITY
NAVAL AIR WARFARE CENTER - WEAPONS DIVISION
NAVAL AIR WARFARE CENTER - AIRCRAFT DIVISION
NAVAL UNDERSEA WARFARE CENTER DIVISION, NEWPORT
PACIFIC MISSILE RANGE FACILITY
NAVAL UNDERSEA WARFARE CENTER DIVISION, KEYPORT

30th SPACE WING
45th SPACE WING
AIR FORCE FLIGHT TEST CENTER
AIR FORCE DEVELOPMENT TEST CENTER
AIR WARFARE CENTER
ARNOLD ENGINEERING DEVELOPMENT CENTER
GOLDWATER RANGE
UTAH TEST AND TRAINING RANGE

DISTRIBUTION A: APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION IS UNLIMITED

19981027 026

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE SEPTEMBER 1998	3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
STANDARD PROTOCOL FOR REAL-TIME COMPUTER TO COMPUTER DATA EXCHANGE

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DATA REDUCTION AND COMPUTER GROUP
RANGE COMMANDERS COUNCIL
WHITE SANDS MISSILE RANGE, NM 88002-5110

8. PERFORMING ORGANIZATION REPORT NUMBER
IRIG STANDARD 168-98

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
STEWS-RCC
RANGE COMMANDERS COUNCIL
WHITE SANDS MISSILE RANGE, NM 88002-5110

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
SAME AS BLOCK 8

11. SUPPLEMENTARY NOTES

NEW DOCUMENT

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document provides a standard protocol for accomplishing real-time interrange exchange of processed data between range computers. The intent of this exchange is to allow monitoring and control of all phases of a test at the appropriately prepared host range mission control while using unique facilities, instrumentation, and capabilities at other test range locations for various phases of the test. Tests that transition from one range to another (e.g. space shuttle, cruise missile) could transition to this standard as a means of providing information and compatibility of formats from range to range and project to project.

14. SUBJECT TERMS
data exchange, real-time data

15. NUMBER OF PAGES
73

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
unclassified	unclassified	unclassified	none

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18 298-102

USAPPC V1.00

STANDARD 168-98

STANDARD PROTOCOL FOR REAL-TIME COMPUTER
TO COMPUTER DATA EXCHANGE

SEPTEMBER 1998

Prepared by

DATA REDUCTION AND COMPUTER GROUP
RANGE COMMANDERS COUNCIL

Published by

Secretariat
Range Commanders Council
U.S. Army White Sands Missile Range
New Mexico 88002-5110

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1.0

Introduction

As DOD pushes towards consolidation, and test/training ranges increase their efforts to avoid unnecessary duplication of capabilities, many opportunities will arise for range users to conduct the various phases of their tests distributed over a variety of ranges and facilities. However, conducting tests effectively and efficiently in this mode will require increased commonality of real-time data handling procedures and the successful completion of plans for real-time inter-range data communications systems. These and related issues are currently being addressed through broad based standards organizations such as SISO (Simulation Interoperability Standards Organization) and the many Joint projects such as TENA (Test/training Enabling Architecture), JSIMS (Joint SIMulation System), JTCTS (Joint Tactical Combat Training System), STOW (Simulated Theater Of War), and JADS Joint Advanced Distributed Simulation), to name a few. HLA (High Level Architecture) is being offered through SISO and mandated by the Defense Undersecretary for Acquisition as the standard for Defense simulation interoperability. Also, commercial standards such as CORBA (Common Object Resource Broker Agent), Microsoft's DCOM (Distributed Common Object Model) and related ActiveX family of products, and other publish/subscribe architectures are being successfully promoted to provide distributed comr ·· ng solutions. There is a continuing need to provide useful, current, and hopefully compa· ·· range software reuse and interoperability standards as technologies evolve and issues get resolved.

The unique world of test/training ranges can be considered simulation using live entities with growing reliance on adding constructive and virtual simulation to mold the future paradigm. Meeting the requirements of more complex testing scenarios, combined testing and training, and Joint exercises demands that the range assets be inter-operable using broad community standards. The standard offered in this document represents a first attempt at providing that interoperability as an interim measure. It is believed that this standard will evolve to embrace HLA and TENA as they become viable architectures for the DOD Test and Training range community.

This document provides a standard protocol for accomplishing real-time inter-range exchange of processed data between range computers. The intent of this exchange is to allow monitoring and control of all phases of a test at the appropriately prepared host range mission control while using unique facilities, instrumentation, and capabilities at other test range locations for the various phases of the test. Tests that transition from one range to the other (e.g. space shuttle, cruise missile) could transition to this standard as a means of providing information and compatibility of formats from range to range and project to project.

This standard not only defines a universal format for transfer for various types of real-time data, it also specifies the behavior associated with the software in the computers at the ends of the data exchange. It is an extendable standard where additional standardized formats can be accommodated when they become available.

RCC (Range Commanders Council) member ranges should use this document and the referenced standards as guidelines for establishing real-time inter-range data exchange capabilities. Evolution of existing real-time data handling procedures should be directed towards compatibility with this data exchange standard and the adoption of the universal data formats where specified for a given data type.

CHAPTER 2.0

Overview

The protocol defined by this standard is based on a *client-server* model, characteristic of many contemporary network applications. Figure 2-1 illustrates the concept. As shown in the diagram, there are two distinct software applications (the server and the client) running in separate range computers. The range computers are typically at two different ranges. In this approach, the server remains vigilantly watchful for requests from clients. The client, based upon operator action or some other unspecified event on the user side, initiates the data transfer session by sending a request to the server. The server is assumed to source the real-time data and the client is assumed to sink the data. That is, the real-time data flows in one direction only (server to client). Two-way exchange of control data occurs between the two for the purposes of setting up and controlling the real-time transfer but the real-time data is passed in one direction only[1].
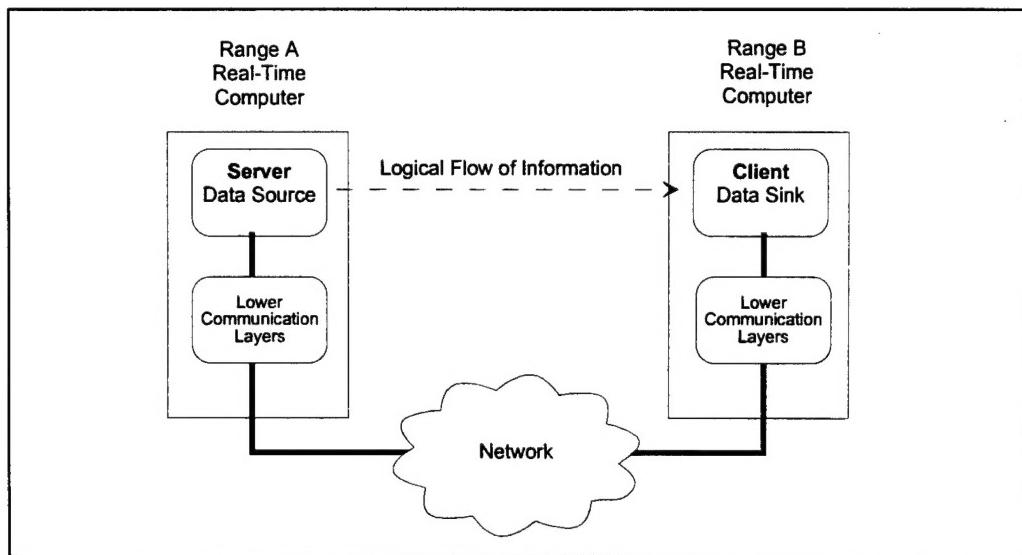
Figure 2-1. Client-Server Architecture.

These software applications communicate by passing messages to each other via the underlying network software and hardware. There is a *logical* communication between the applications shown in the diagram. The actual transfer of messages occurs transparently to the application software via well-known off-the-shelf networking concepts. This standard defines the *logical* exchange of messages between the client and the server. Generically, the client and server are referred to as Application layer *Protocol Entities* and the messages exchanged between them are called Application Layer *Protocol Data Units* or PDUs. The underlying

---

[1] There may be cases where it is desirous to pass real-time data in both directions between two range computers. This has been considered and is described later in the document.

communication layers are based on the popular Internet protocol suite and widely supported sockets model.

Application layer PDUs may be multiplexed or bundled with PDUs from other sources as they are passed through the network. We delegate this to the underlying layers, assuming that a PDU handed to the lower layer will be delivered intact as a unit to the protocol entity on the other side. Though we specify network standards to be used in the lower communication layers, we are not concerned with the implementation details at those layers since these are usually embedded in contemporary operating systems.

The PDUs which flow between the Protocol Entities, together with a description of the state behavior of the applications, are known as a *protocol*. It is important to keep in mind the particular protocol entity being described when reading this document since a protocol specifies the behavior of a specific protocol entity in its interactions with other protocol entities.

There are a number of examples of client server architectures. File Transfer Protocol (FTP) and TELNET protocols are two popular examples. The widespread Internet exploration software, World Wide Web (WWW), is also based on this concept – thought it may not be as evident to the user as the first two examples.

For real-time data exchange between range computers, the server will typically provide real-time data from one or more exercises or tests that are on-going at a particular range. The client can reside in any computer that has network connectivity to the server. Clients initiate the exchange according to an agreed upon protocol by sending messages to a specific port address (that of the server) on a specific machine. The server responds, assuming the client meets authentication requirements, by setting up the real-time data source. To keep the protocol as simple as possible it is assumed that the client has knowledge of the data availability at any given server. This information is passed to the client in some means other than via the protocol. Examples might be telephone conversations, Email or other typical coordination activities between human beings.

Any number of data types can be accommodated with the IRIG STD 168-98 protocol. The types of data include all real-time data that may be sent from one computer to another. Examples are radar data, telemetry data, and digitized voice. There may also be a number of each of these available at the same time from a given range computer. Multiple radars may be sending data simultaneously. More than one telemetry stream, as well as multiple voice channels, may be available at the same time. This gives rise to the diagram shown in Figure 2-2. Here we introduce the notion of multiple simultaneous real-time data streams flowing between range computers. These data streams may be all related to the same mission or there may be multiple simultaneous missions. The architecture is meant to accommodate both situations. Figure 2-2 shows data flowing between a single source computer and a single destination computer. Many situations call for real-time data to flow to multiple destinations simultaneously.
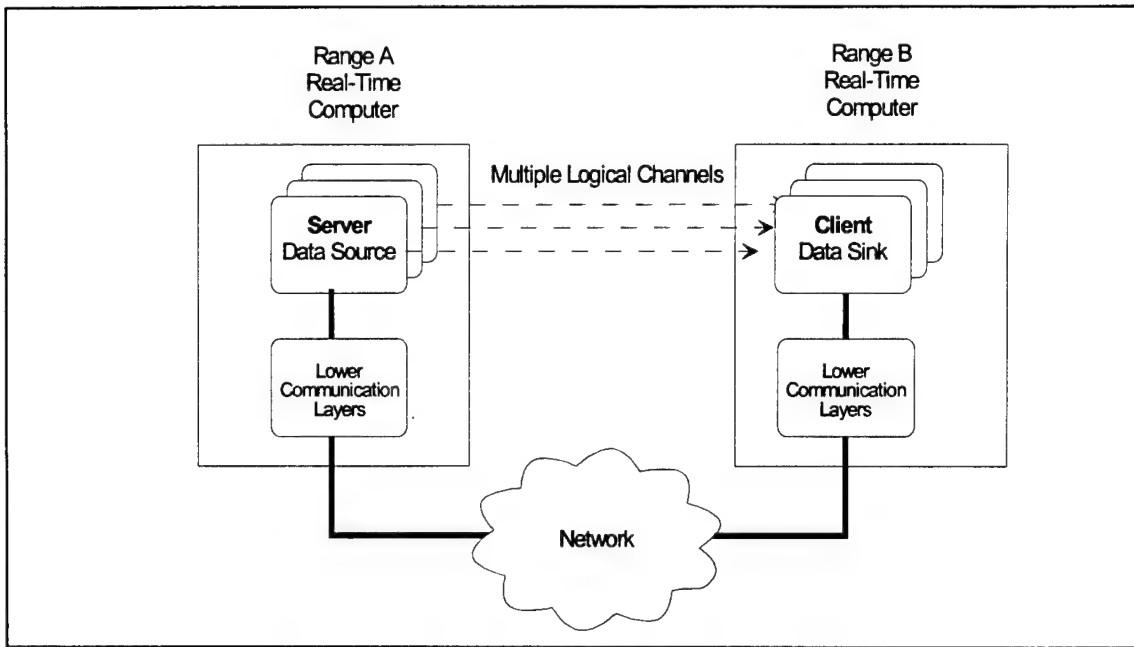
Figure 2-2. There Can Be Multiple Data Streams.

Figure 2-3 Illustrates the situation where multiple data streams are flowing to more than one range computer at the same time. This multiple simultaneous data distribution situation is common for missions that span large geographical areas or situations where the data is of interest to a wide audience.
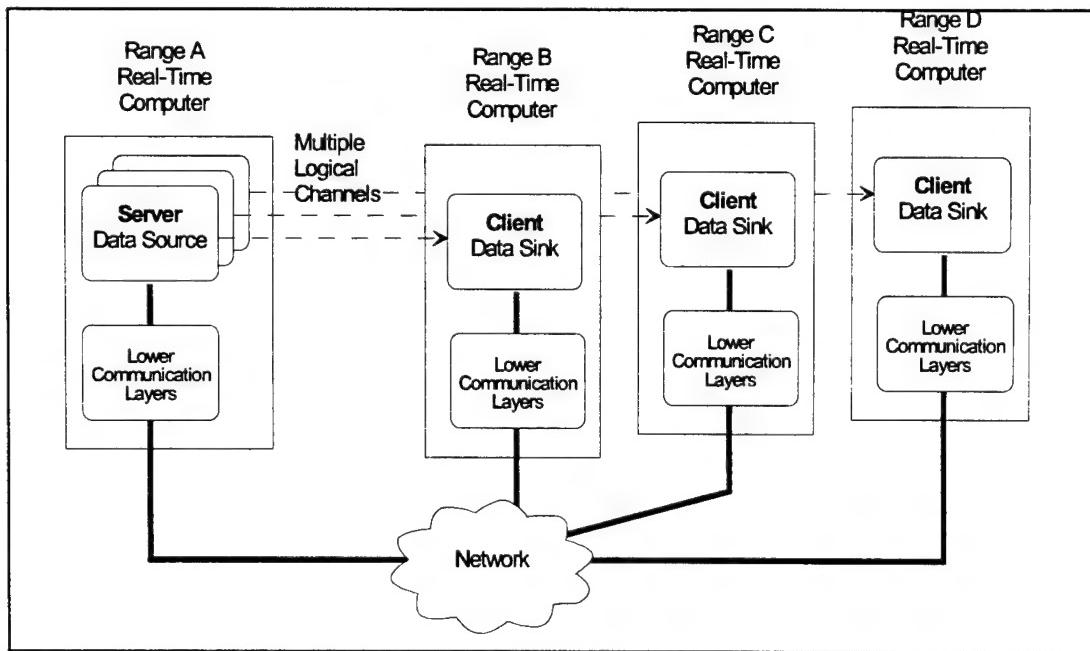
Figure 2-3. Data Can Go To More Than One Computer.

In other situations, data from a number of sources may be of interest at a particular location and consequently, there may be a need to accommodate data from multiple sources. This situation is indicated in Figure 2-4. Here range computers A, B, and C are sending data to range computer D.
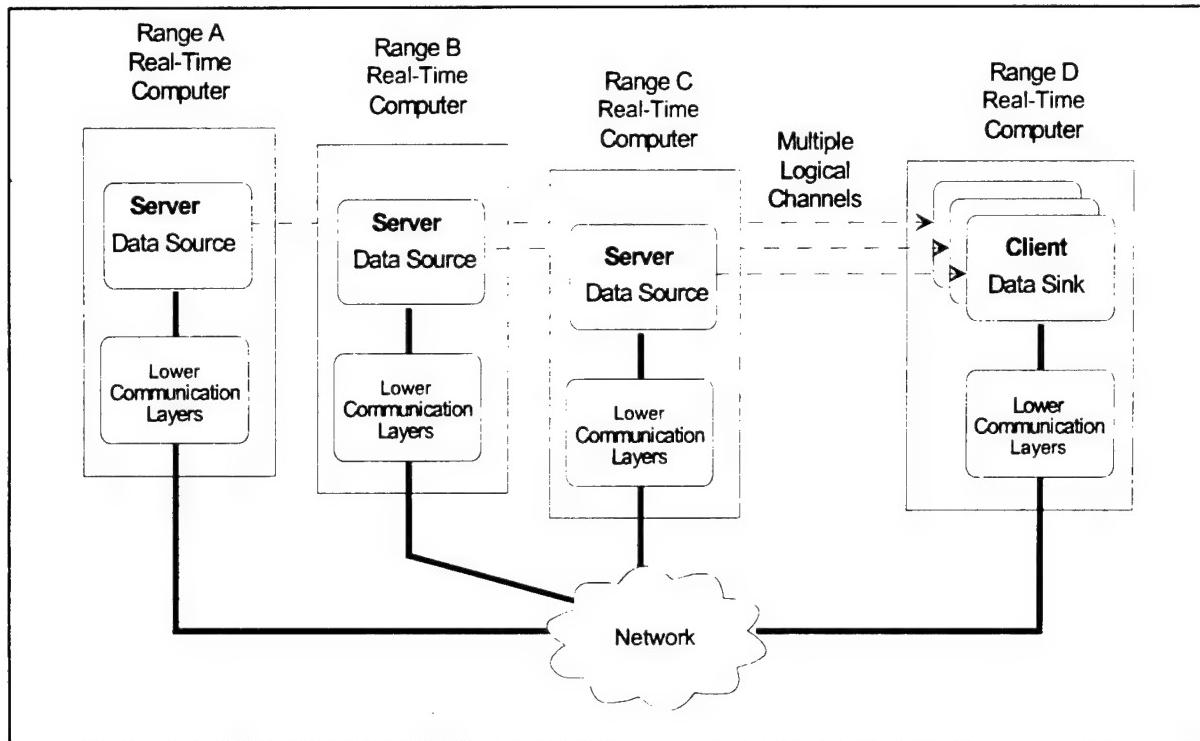
Figure 2-4. Multiple Sources Can Send Data.

In most cases, real-time data flows from a range where a mission is being conducted to one or more ranges where the data is of interest. There may be situations where the real-time data flows in both directions simultaneously. This situation can be handled by the architecture as shown in Figure 2-5. As shown in this example, range A is sending data to range B and at the same time range B is sending data to range A. The architecture handles this situation by including Client and Server protocol entities in each of the two range computers.

The above discussions have presented a number of sample configurations. There are a number of other possible configurations based on the client-server architecture, but those presented above represent the ones of greatest interest to the range community.
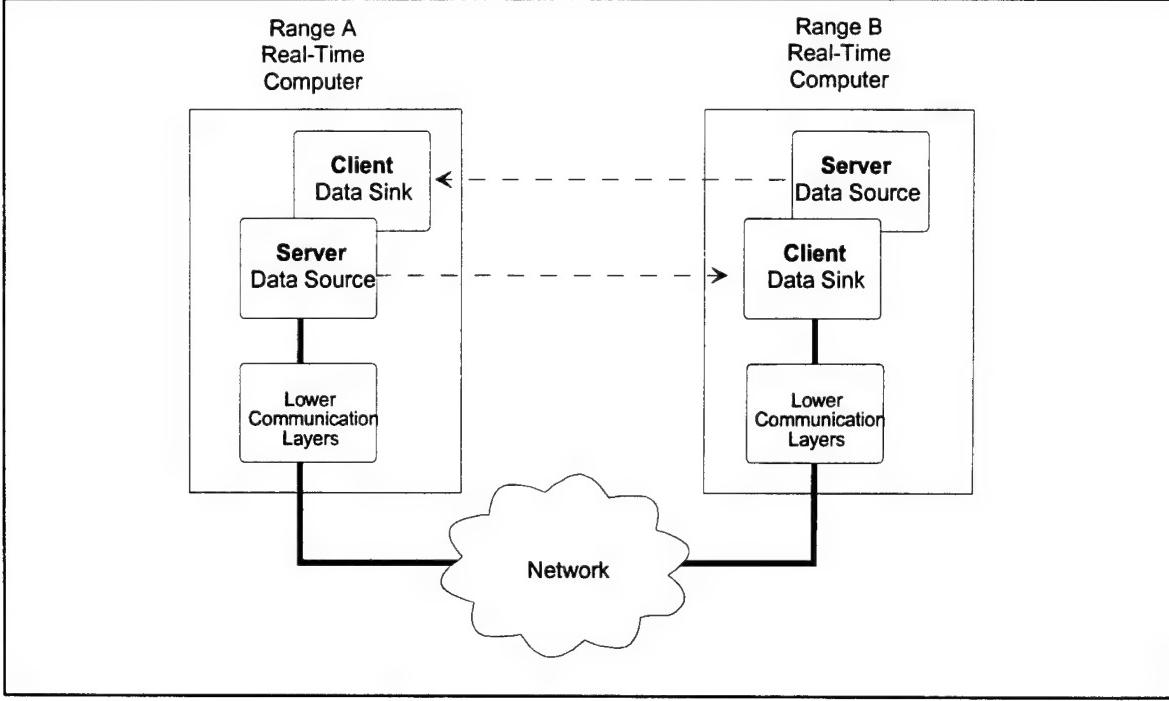
Figure 2-5. Real-Time Data Can Flow In Both Directions.

Figure 2-6 shows a model that indicates the concept that the Client and Server protocol entities need not be associated with a particular range computer. They are, in our model, independent of the host computer. In fact, both can reside in the same computer. In addition, there may be a number of instances of each in a given network or in a given computer
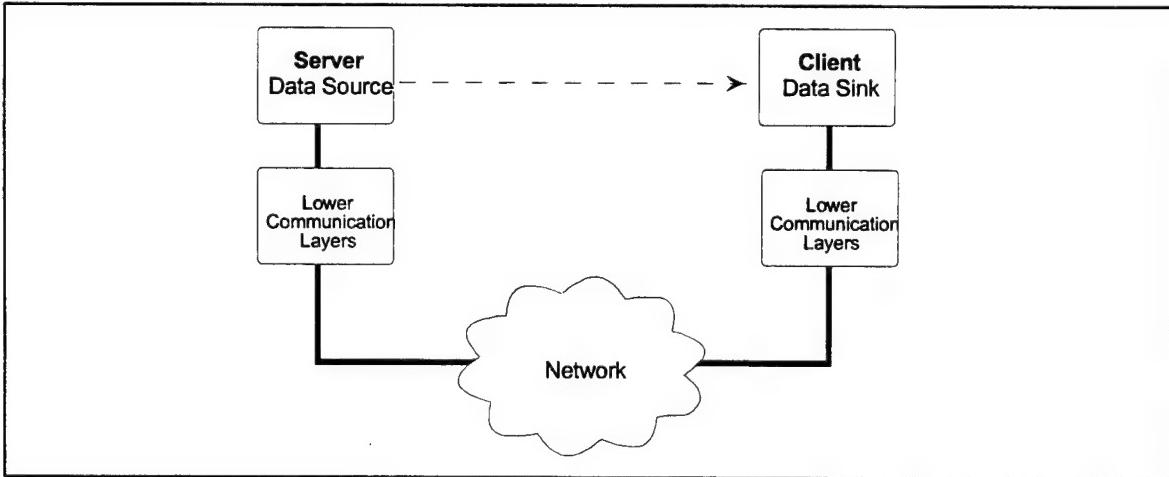
Figure 2-6. Architectural Model.

In the Internet suite of protocols there are two popular transport layers, either of which can be used. The first, Transmission Control Protocol (TCP), Reference 1, has end to end delivery reliability (at the potential expense of added latency) and the second, User Datagram Protocol (UDP), Reference 2, has low latency but does not try to recover lost messages. For real-time data one usually uses UDP and for status and control one uses TCP. However, the real-time data transfer drives our needs and, though a small amount of additional processing is needed in the application layer, UDP was chosen as the transport layer for both control and real-time data transfer.

As shown in Figure 2-7, a real-time data transfer session consists of three main phases. It begins with an Initialization and Setup phase where information is exchanged between the Client and the Server necessary to setup the real-time transfer. The process is initiated with a request from the Client to the Server. If all goes well and the Server can comply with the request, the session progresses to the next phase where real-time data is transferred. Data transfer occurs between the Server and the Client as essentially a one-way transfer. Finally, after mission completion, the session is terminated with a post mission data exchange. This post mission data exchange contains information relating to the mission that would not normally be transferred during real-time. It also contains resource utilization information, which can be used for accounting purposes.

Figure 2-7 is a Message Sequence Chart (MSC)[2] overview of the process. The following sections describe this process in detail.
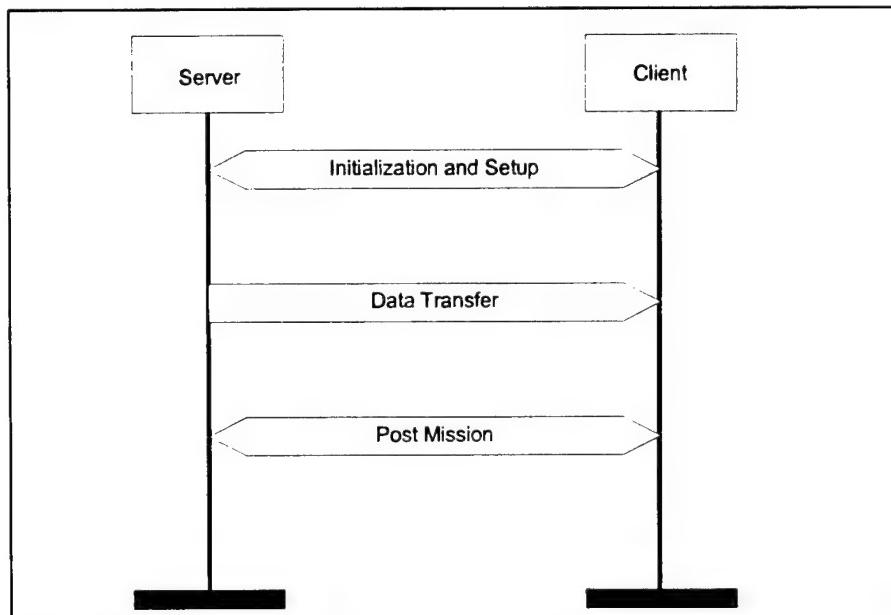
Figure 2-7. MSC Overview.

---

[2] Message Sequence Charts are described in section 16.1. They should be read as a time sequence of events beginning from the top and proceeding down the chart.

CHAPTER 3.0

Normal Sequence

We begin by looking at the details of a normal session. This is shown in Figure 3-1. The process begins with some non-specified (N/S) action on the user side. The first arrow originating from the outer rectangle in the diagram indicates this. Next, a "Subscribe" PDU is sent from the Client Protocol Entity to the Server Protocol Entity. Here the Client must know the Internet Protocol (IP) address of the machine in which the Server resides. In some situations there may be only one server on the network, but the architecture allows any number of servers. The server listens on a specific port, which must be known to the client as well. The Client sends its messages to that port. In this case the "Subscribe" PDU includes sufficient information for the server to set up a real-time data transfer session and does so immediately. A positive response to the Client Control is in the form of an "Accept" PDU, which the Client uses to setup for the data transfer phase and begin accepting data from the Server.
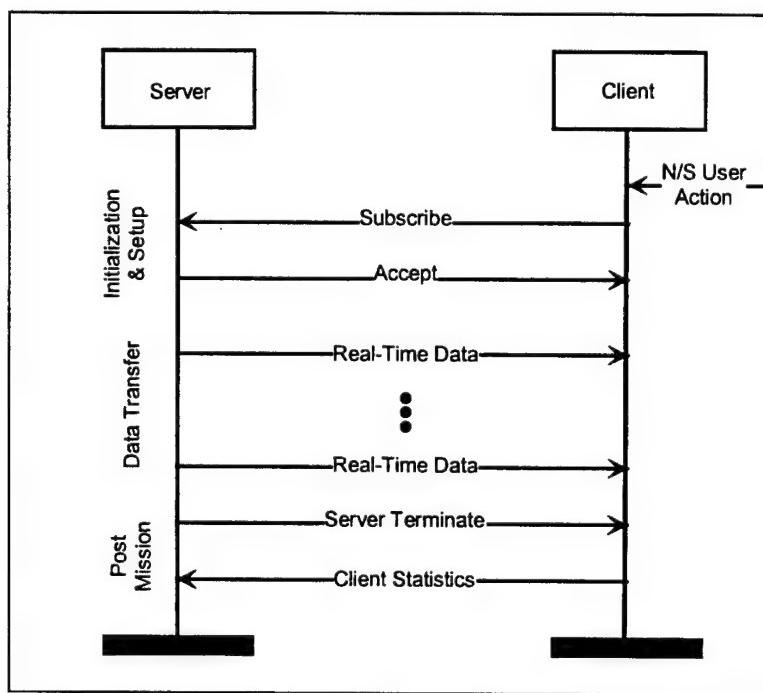
Figure 3-1. Normal Sequence – Server Terminates.

The initialization and setup phase establishes the particular parameters needed for the data transfer. It serves several purposes. These are:

- Authenticate the Client – is this Client authorized to receive the requested data?
- Select the specific mission – more than one could be available from the server
- Select the specific data type (radar, telemetry, etc.) for a specified mission
- Select the specific data format for the specified data type

For a typical session, the real-time data will continue to be sent from the Server to the Client until mission termination. The Server, being first aware of this event, notifies the Client with a "Server Terminate" PDU which contains post mission statistics from the Server. The Client responds with a "Client Statistics" PDU which contains post mission statistics from the Client. In this way both the Server and the Client end up with statistical information concerning the session as well as any other post mission data, such as resource utilization, which might be of use in the final accounting of the session.

It may not always be the Server that terminates the session. The Client may also decide to terminate and does so with a "Client Terminate" PDU as is shown in Figure 3-2. This sequence is identical to that shown above in Figure 3-1 with the exception that the termination originates from the Client side. Though we list both of these as normal sequences, either could represent a normal sequence with an abnormal termination, such as occurs when the user on either side terminates the session. The Termination PDUs contain a code, which indicates the reason for termination. This information can be made available for user examination or just logged.
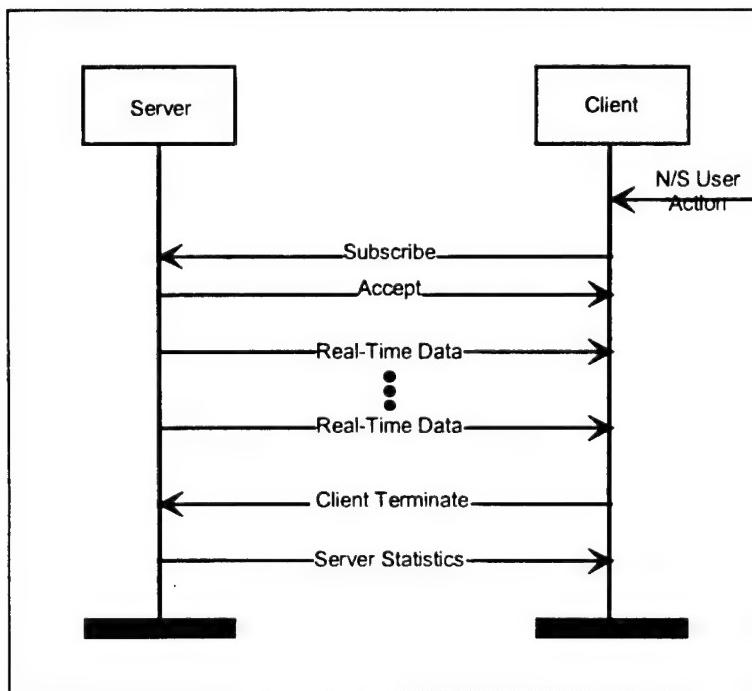
Figure 3-2. Normal Sequence Client Terminate.

CHAPTER 4.0

Abnormal Setup

The following sections address abnormal situations where PDUs are lost or corrupted. The underlying UDP transport layer includes error detection check bits to determine if the PDU was corrupted prior to being received. Since, corrupted PDUs are likely to contain misleading or worthless information UDP discards them. Thus we treat corrupted and lost PDUs the same and refer to both simply as corrupted on our charts. A dashed line which doesn't quite reach to the receiver indicates that the situation.

Figure 4-1 indicates the situation when the "Subscribe" PDU is lost or corrupted. On the right side of the chart is a small symbol that represents an hourglass. This denotes a timer. In this case, the timer is T1, which is specified in the protocol timer values in Table 12-1. T1 represents more than enough time to send a PDU and to receive a response. As shown in the chart, if no response is received by the time T1 expires, the Client will retransmit the "Subscribe" PDU. This process will continue until the Server responds or until a maximum number of retries has occurred. In the latter case it is assumed that the Server cannot be reached. This "Server Not Responding" case is shown in Figure 4-2.
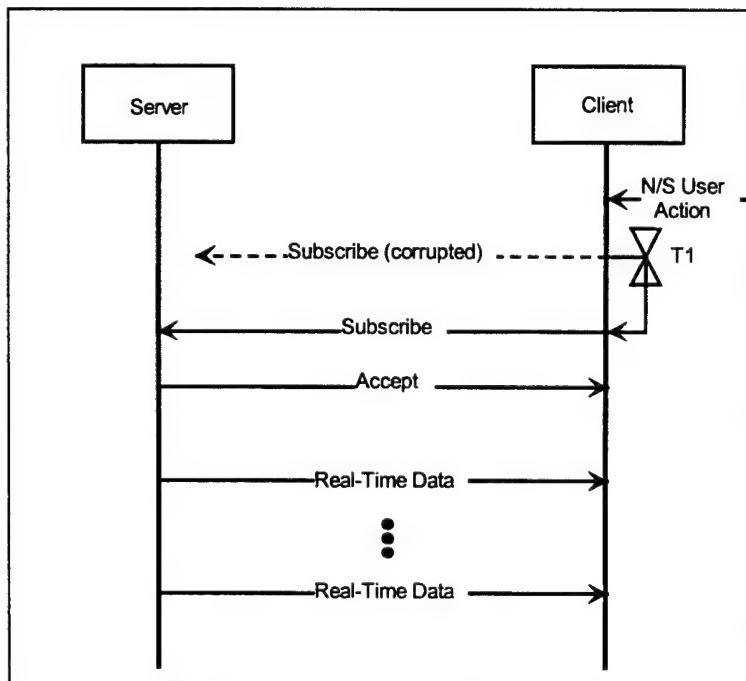
Figure 4-1. Corrupted "Subscribe" PDU.
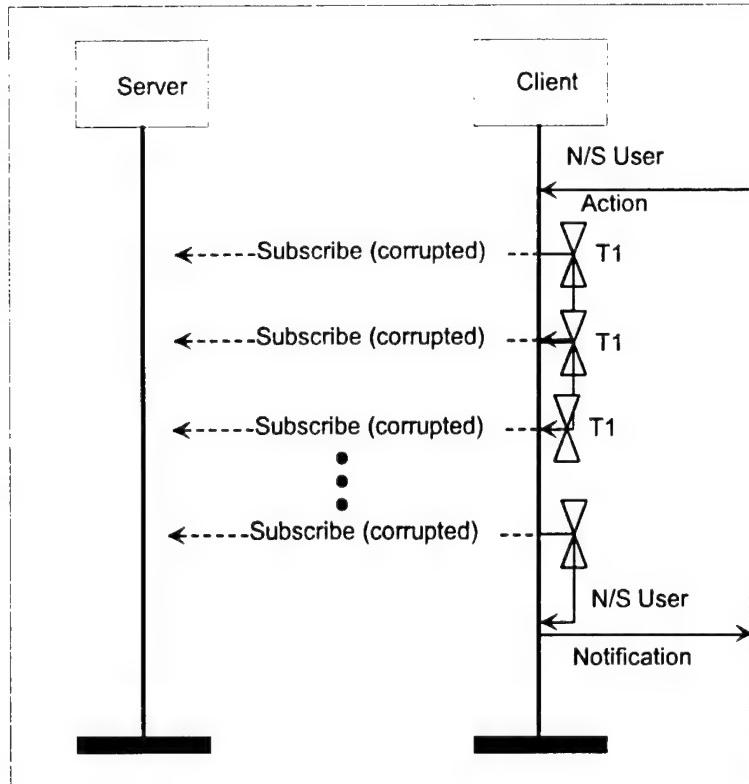
Figure 4-2. Server Not Responding.

A slight variation of the above sequences is when the "Subscribe" PDU is received correctly but the response (in this case, the "Accept" PDU) is corrupted. The Client responds in the same way to the corruption of an "Accept" PDU as it does to the corruption of a "Subscribe" PDU. This corrupted "Accept" PDU case is shown in Figure 4-3. The client retries until an "Accept" PDU is received regardless of which PDU was corrupted. It is the response to the initial "Subscribe" which the Client is awaiting and if this is not received (regardless of the reason) timer T1 expires and the Client tries again.
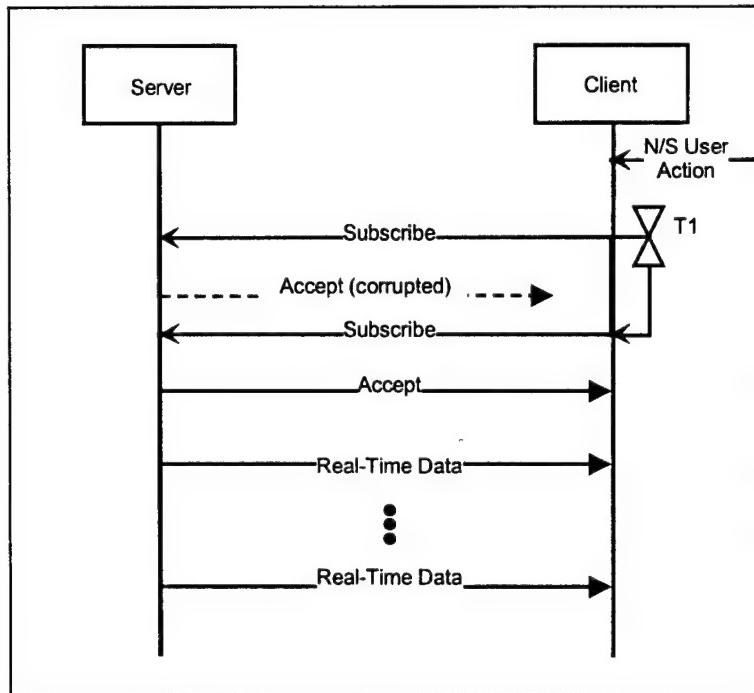
Figure 4-3. Corrupted "Accept" PDU.

If one examines Figure 4-3 closely, it can be seen that there is a potential problem when the "Accept" PDU is corrupted. The Server, having sent the "Accept" PDU, is now ready to begin sending real-time data; however, the Client, having not received the "Accept" PDU, is not ready to receive real-time data. This situation is handled by requiring the Server to wait before beginning the transmission of real-time data. This "Server Waits Before Data Transfer Phase" case is shown in Figure 4-4. Here one can see that the timer T2 is started when the Server sends the first "Accept" PDU that is corrupted. T2 is reset when the second "Subscribe" PDU is received and the second "Accept" is transmitted. Then upon expiration of T2 without receipt of another "Subscribe" PDU the Server begins sending "Real-Time Data" PDUs.
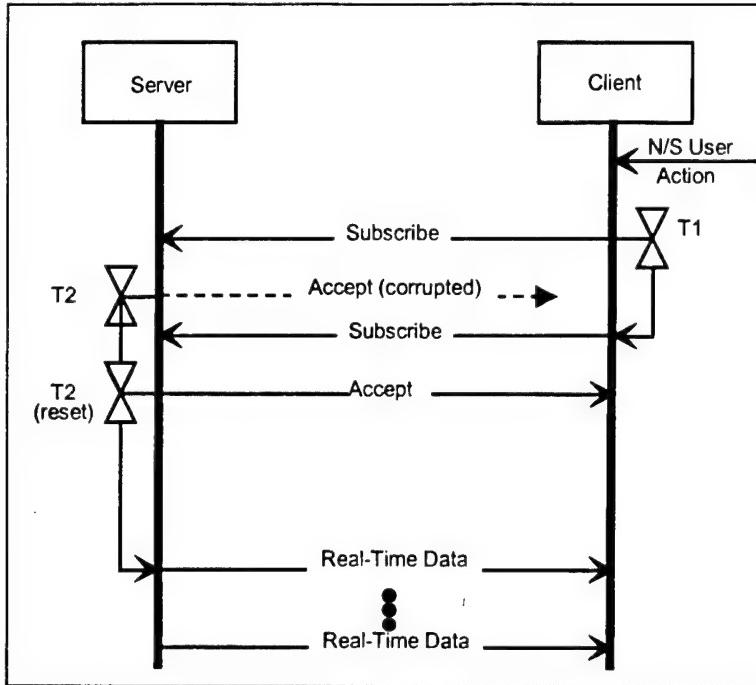
Figure 4-4. Server Waits Before Data Transfer Phase.

It is possible that the second "Subscribe" PDU in Figure 4-4 is lost or corrupted; however, the T1 timer on the client side will protect this case as well. The client will resend the "Subscribe" PDU if T1 expires before receipt of an "Accept" PDU. This protects against lost of either PDU. The process will continue for a maximum number of retries (specified by R1) or until the PDU exchange is completed successfully. The relationship between the T1, T2 and R1 is an important one. See Tables 12-1 and 12-2 for a definition of this relationship.

All the above show the normal situation where the Server accepts the subscribe request. There is the possibility that the Server will reject this request. This can occur for a variety of reasons such as:

- the Client cannot be Authenticated
- the Client is not Authorized to receive the requested data
- the requested data is not available
- the requested data format is not available

Figure 4-5 illustrates the reject sequences including those situations where PDUs are corrupted during the process.
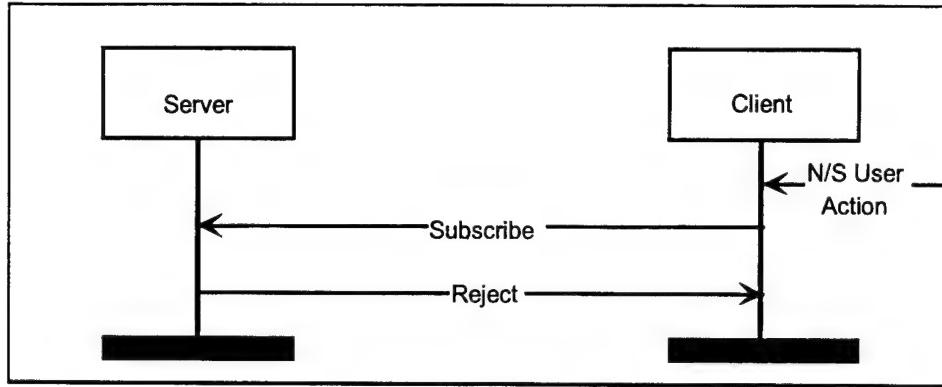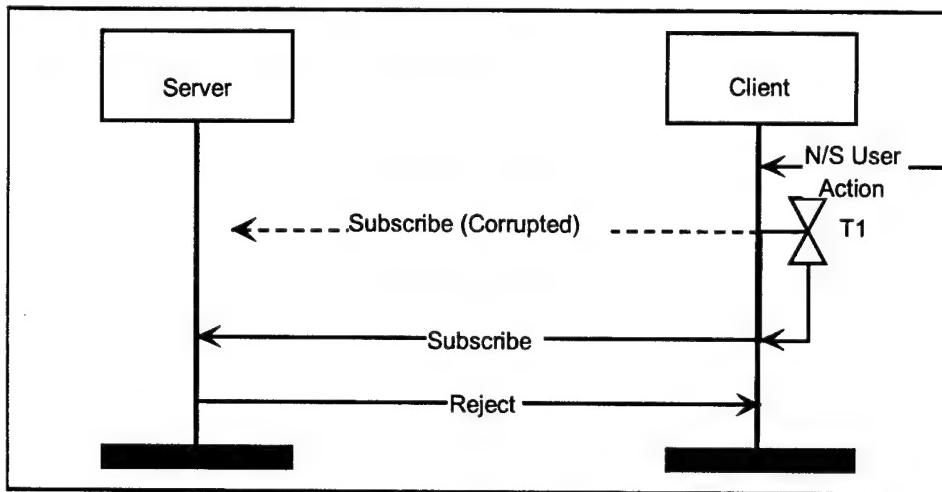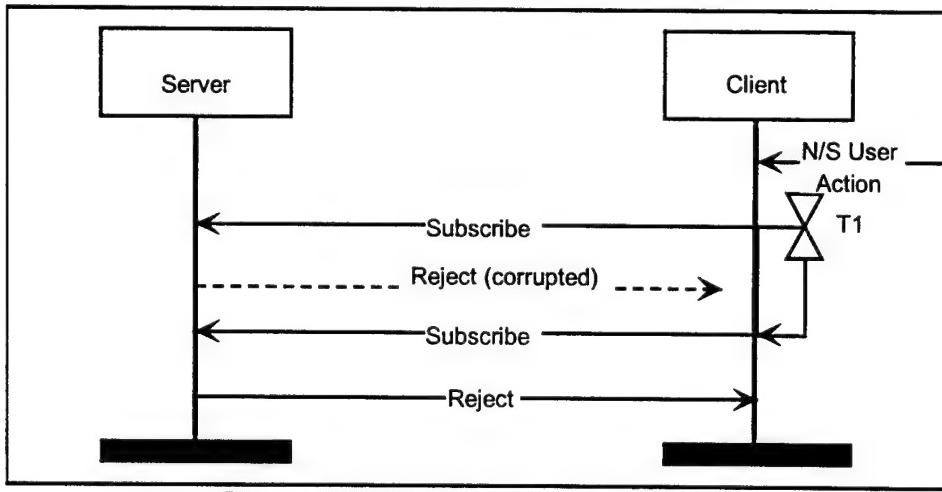
Figure 4-5. Reject Sequences.

CHAPTER 5.0

Data Transfer Phase

We now examine the message sequences that may occur during the data transfer phase. Since we define this as a simple one way transfer of data the message sequence charts are very simple. We still must examine the abnormal situations. Each UDP PDU includes error check bits, which provide the means to detect corrupted messages. Corrupted messages are discarded and thus treated the same as lost messages. We assume that the importance of the next sample of Real-Time Data is more important than trying to recover the lost data. The situation of a Corrupted "Real-Time Data" PDU is shown in Figure 5-1.
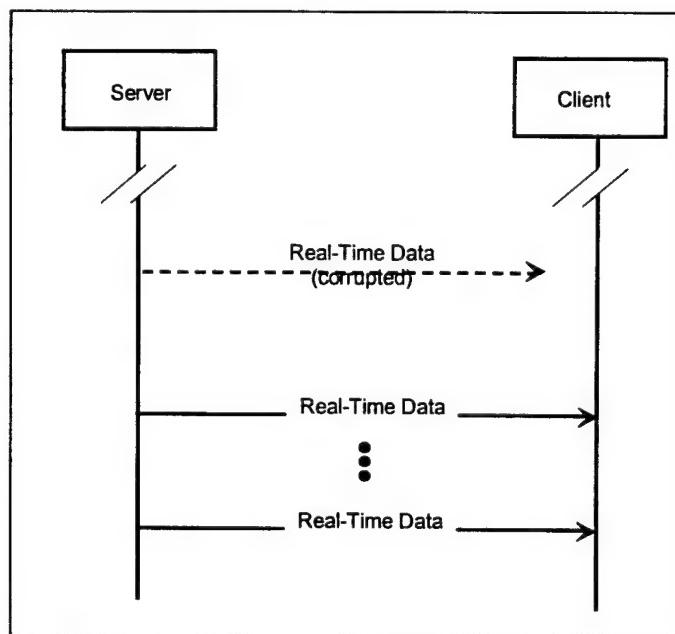
Figure 5-1. Corrupted "Real-Time Data" PDU.

Since we are using UDP as the transport and since UDP does not guarantee that PDUs will be received in proper order, we must make provision for handling PDUs received out of sequence. This occurs most often in networks where the routing through the network is dynamic. In such circumstances, the PDUs may take different paths through the network and in such cases can arrive out of order. We don't expect this to be a significant problem with most data exchange between range computers since the network route between the end points tends to be deterministic. Thus we treat this as a rare occurrence and simply discard PDUs that are received after a more recent one has already been received.

To detect PDU sequence errors and to detect lost PDUs, we include a sequence counter in each PDU. This is an 8-bit counter, which is monotonic and increasing by one (modulo 256) for each PDU in sequence. If a PDU is received with a sequence number which is more than one greater than the most recently received one it is assumed that the PDUs associated with the intervening sequence numbers have been lost. If a PDU is received with a sequence number less than the sequence number of the most recently received PDU it is discarded. The "Real-Time Data" PDU Out of Sequence situation is illustrated in Figure 5-2. Here PDU 006 is received prior to PDU 005. When this occurs, PDU 005 is declared corrupted and then when it is later received it is discarded.
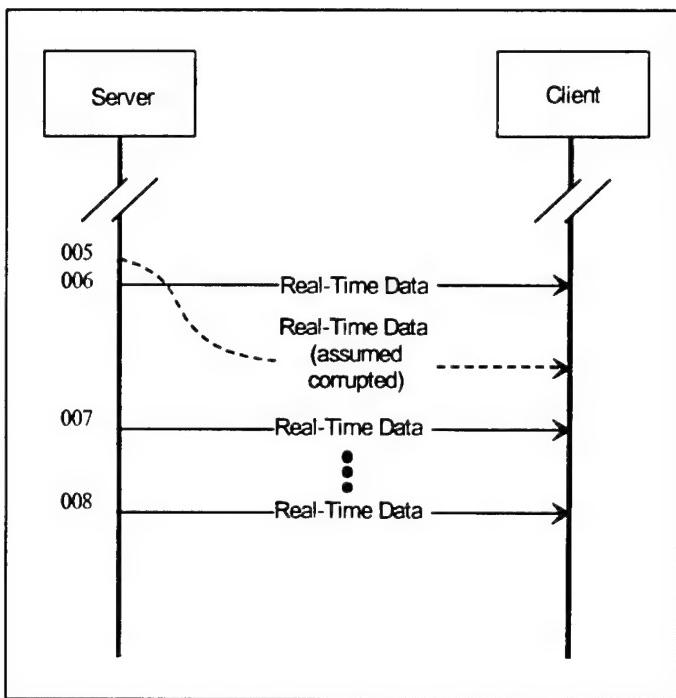
Figure 5-2. "Real-Time Data" PDU Out of Sequence.

To insure that the connection is maintained and that the sequence counter progresses normally, we must deal with situations where the real-time data is not available for long periods of time. This could be indistinguishable from situations where "Real-Time Data" PDUs are available but are being corrupted or lost unless we specify the maximum amount of time between "Real-Time Data" PDUs. If this time expires without receipt of a "Real-Time Data" PDU the Client can determine that data has been lost or worse yet the connection itself has been lost. Thus, we specify that a "Keep-Alive" PDU be transmitted when Real-Time data is not available and the maximum time between "Real-Time Data" PDUs has expired. The concept of "Keep-Alive" PDUs is illustrated in Figure 5-3.
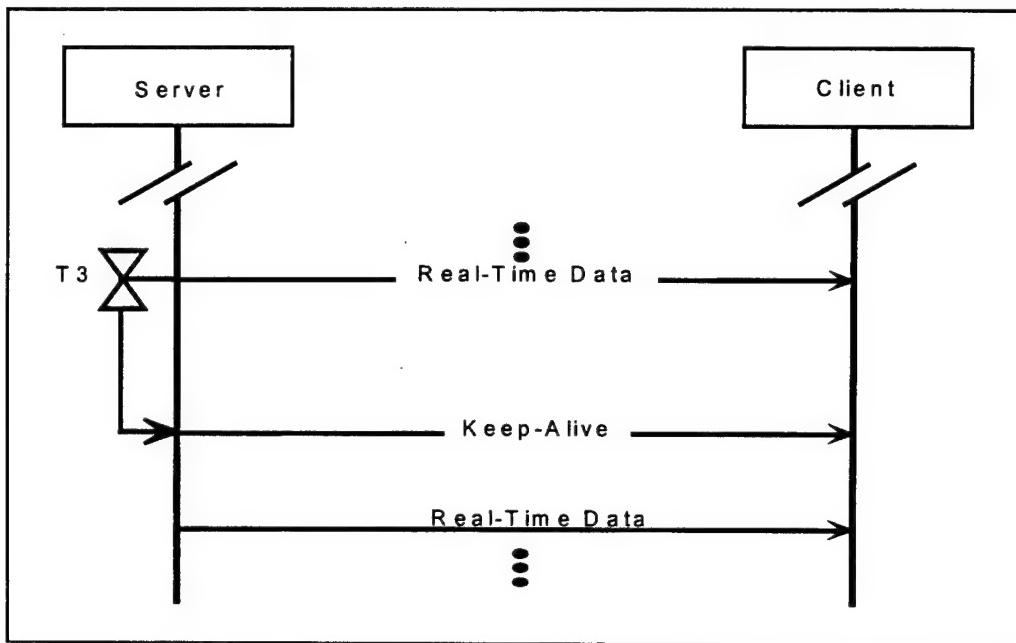
Figure 5-3. "Keep-Alive" PDU Sent When Data is Not Available.

CHAPTER 6.0

Abnormal Terminates

6.1 Abnormal Server Terminates

We now look at abnormal situations associated with termination of the session. Figure 6-1 shows the situation where the Server terminates the session but the "Server Terminate" PDU is corrupted. In this situation, the timer, TI, set by the Server expires without a response from the Client and the Server retransmits the "Server Terminate" PDU.
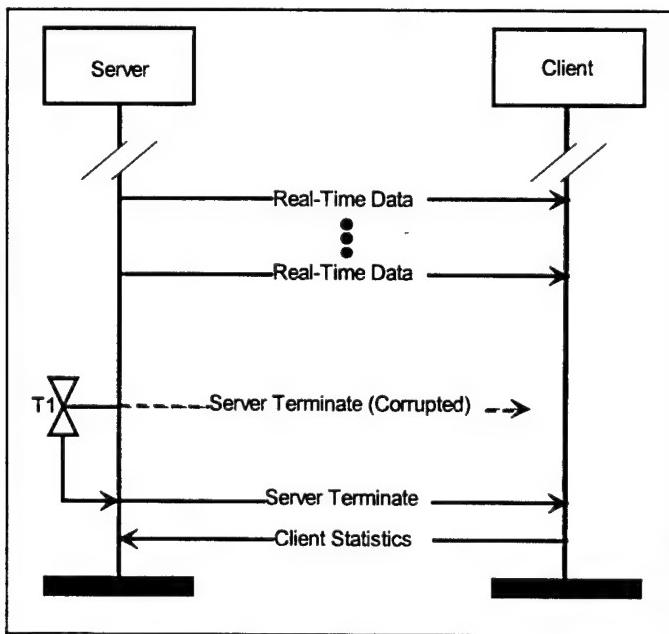
Figure 6-1. Corrupted "Server Terminate" PDU.

Of course the "Client Statistics" PDU could also be corrupted and this is treated in very much the same manner as a corrupted "Server Terminate" which is described above. As shown in Figure 6-2 the Server retransmits the "Server Terminate" until the expected response is received regardless of which PDU is corrupted.
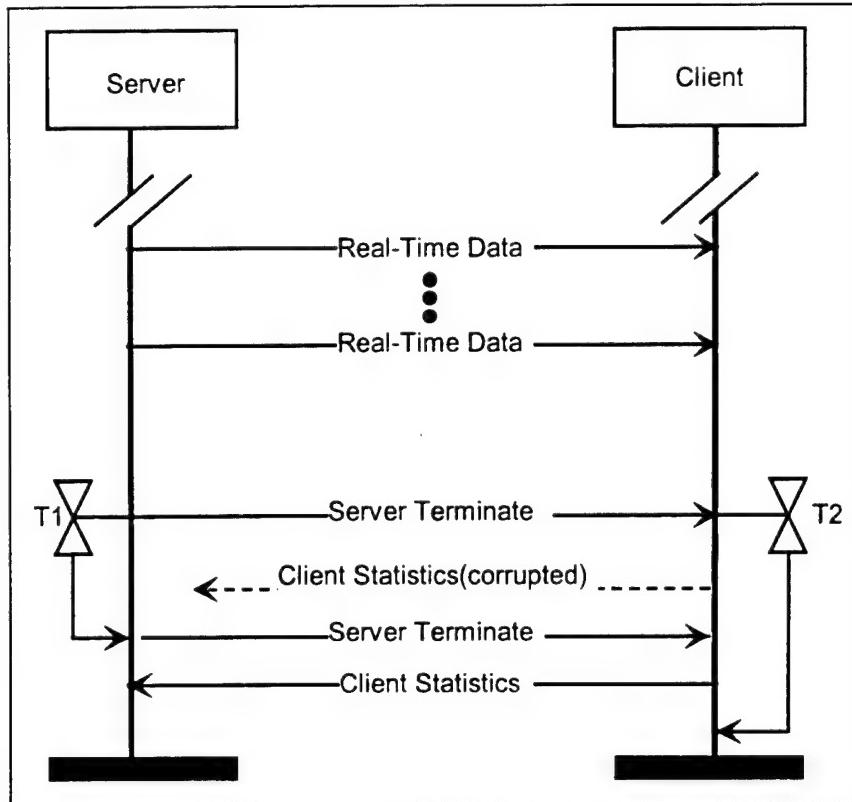
Figure 6-2. Corrupted "Client Statistics" PDU.

It is most unlikely, but nevertheless possible that the Server would not receive any response to the "Server Terminate" PDU. In this case the Server will exhaust its retries and terminate abnormally. Likewise the Client would terminate after timing out as well. This Server Terminate After Retries Exhausted case is shown in Figure 6-3.
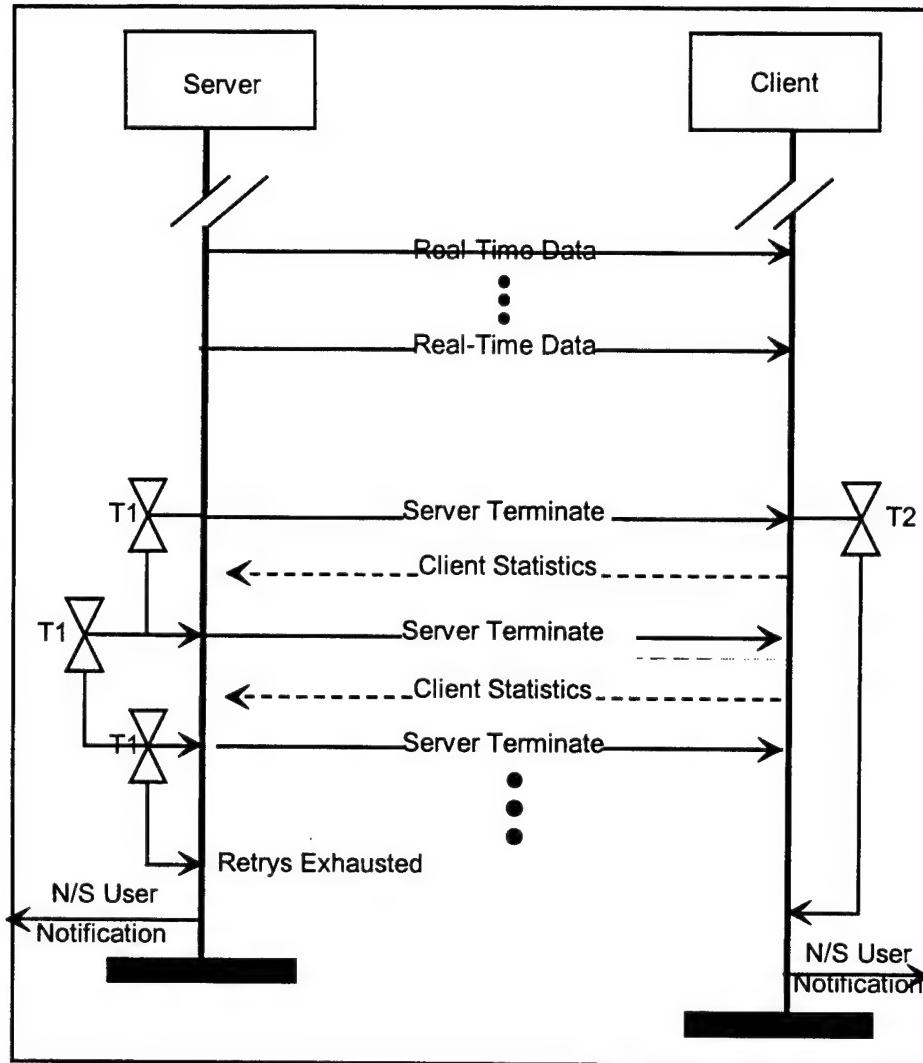
Figure 6-3. Server Terminate After Retries Exhausted.

6.2 Abnormal Client Terminates

In those situations where the Client terminates the session, the methods for handling the corruption of PDUs is very similar to those described above for server-side terminate. Figure 6-4 shows the situation when the "Client Terminate" is corrupted.
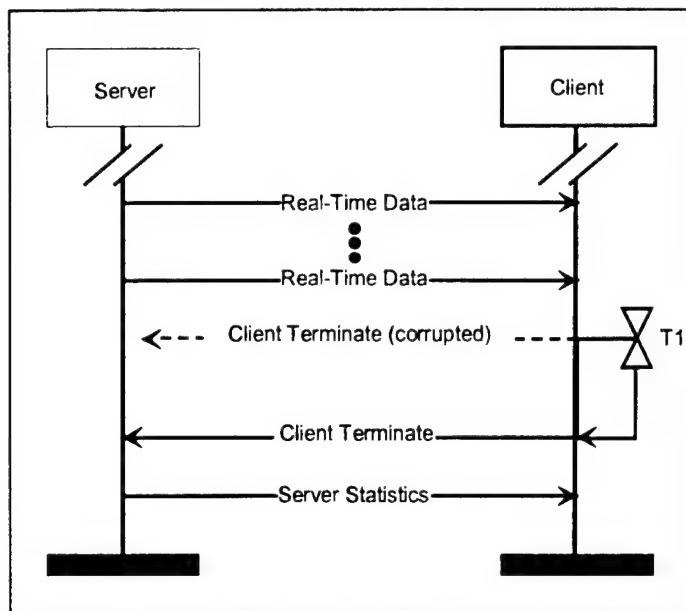
Figure 6-4. Corrupted "Client Terminate".

As illustrated above for terminates from the server side, the client side termination process is protected against loss of the "Server Statistics" PDU by the same timer. No MSC is provided to illustrate this obvious case. However, total failure of the link from the Client side could result in a situation where the Client has exhausted all time outs and retries and thus disconnects without the server being aware of the departed Client. In this case the Server, may become aware of the lost connect (depending on the particular implementation of the underlying communications software) or it may not. In the worse case, the Server will continue to send PDUs (probably "Real-Time Data" PDUs) until the completion of the mission and then go through an abnormal terminate as illustrated in Figure 6-3.

CHAPTER 7.0

Protocol Specification

IRIG 168-98 defines the behavior of the Client Protocol Entity and the Server Protocol Entity using a graphic representation based on the Specification and Description Language/Graphical Representation (SDL/GR).[3]

7.1 <u>Client Behavior Specification</u>

Figures 7-1 through 7-3 display Client Graphic Representation.

---

[3] The SDL/GR is described in Section 16.2 and is a combination of traditional logic flow charts and state transition diagrams.
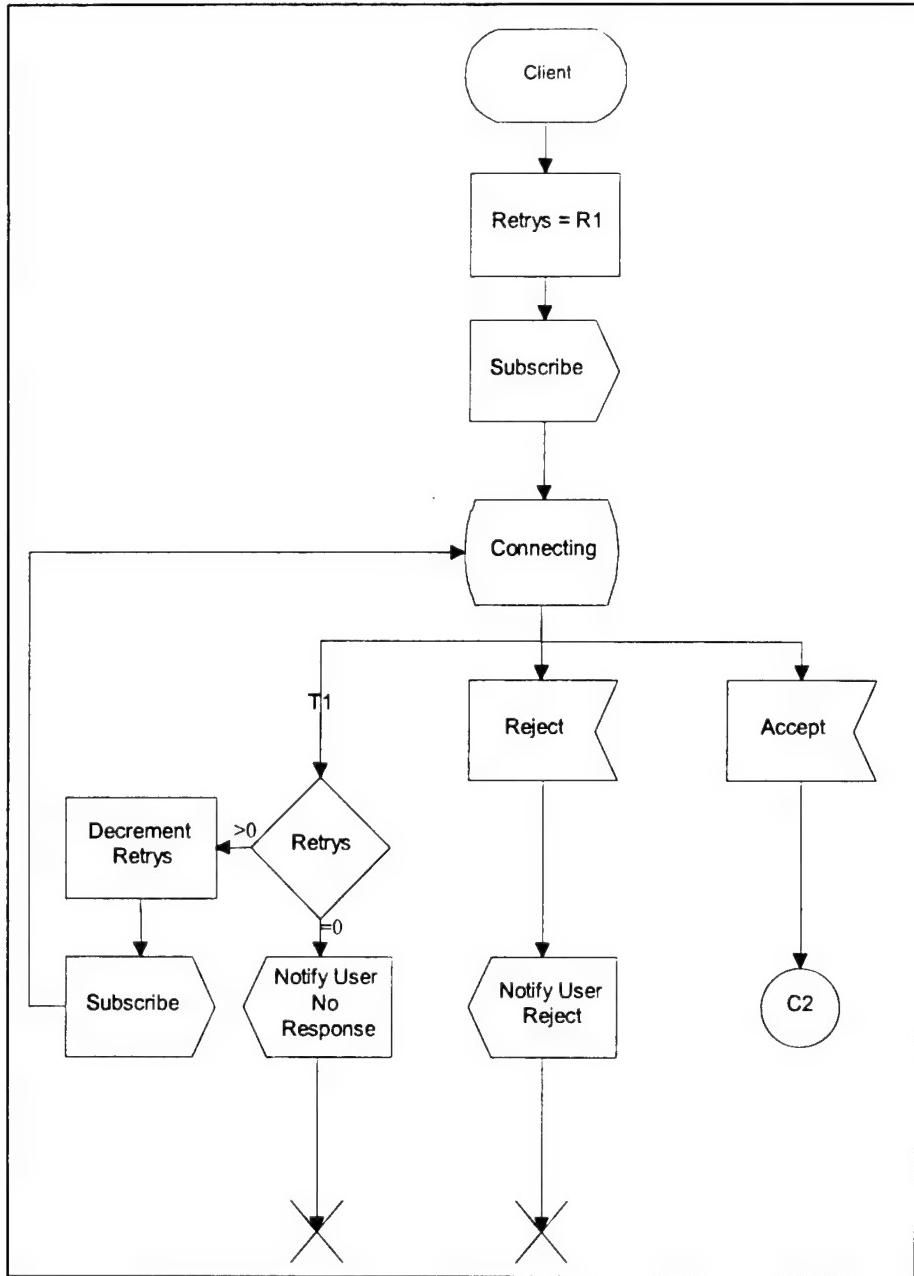
Figure 7-1. Graphic Representation Client (1 of 3).
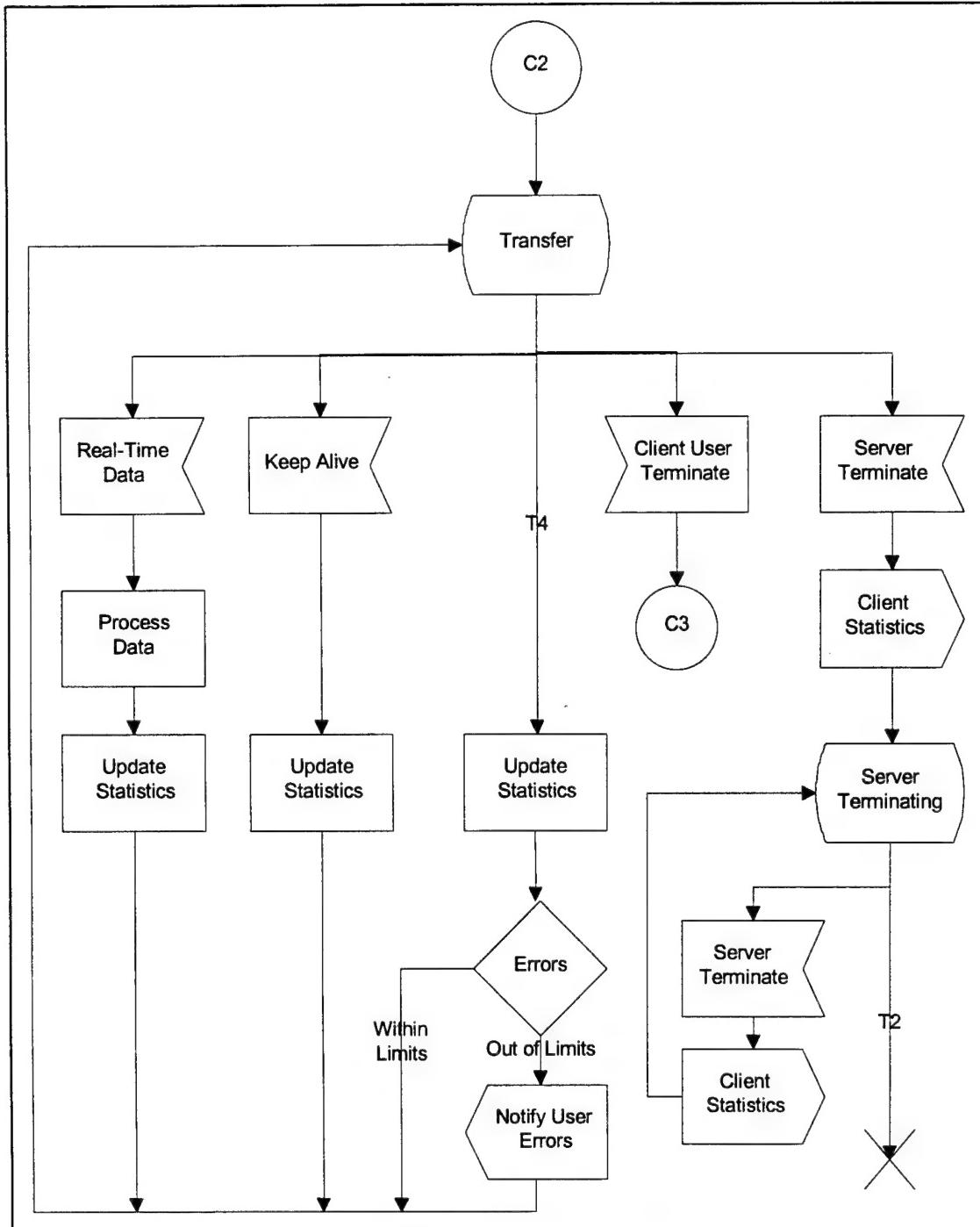
Figure 7-2. Graphic Representation Client (2 of 3).
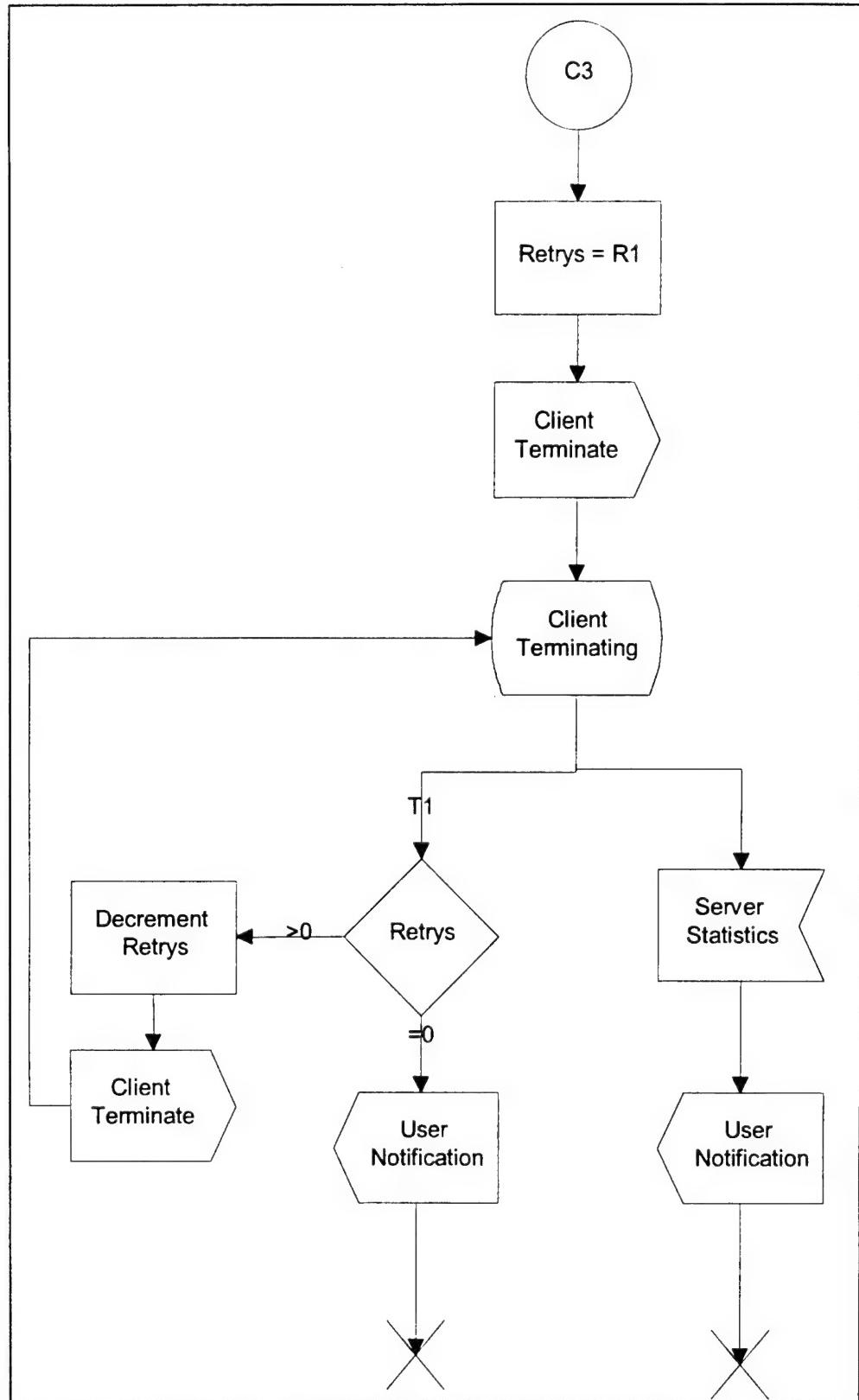
Figure 7-3. Graphic Representation Client (3 of 3).

7.2 Server Behavior Specification
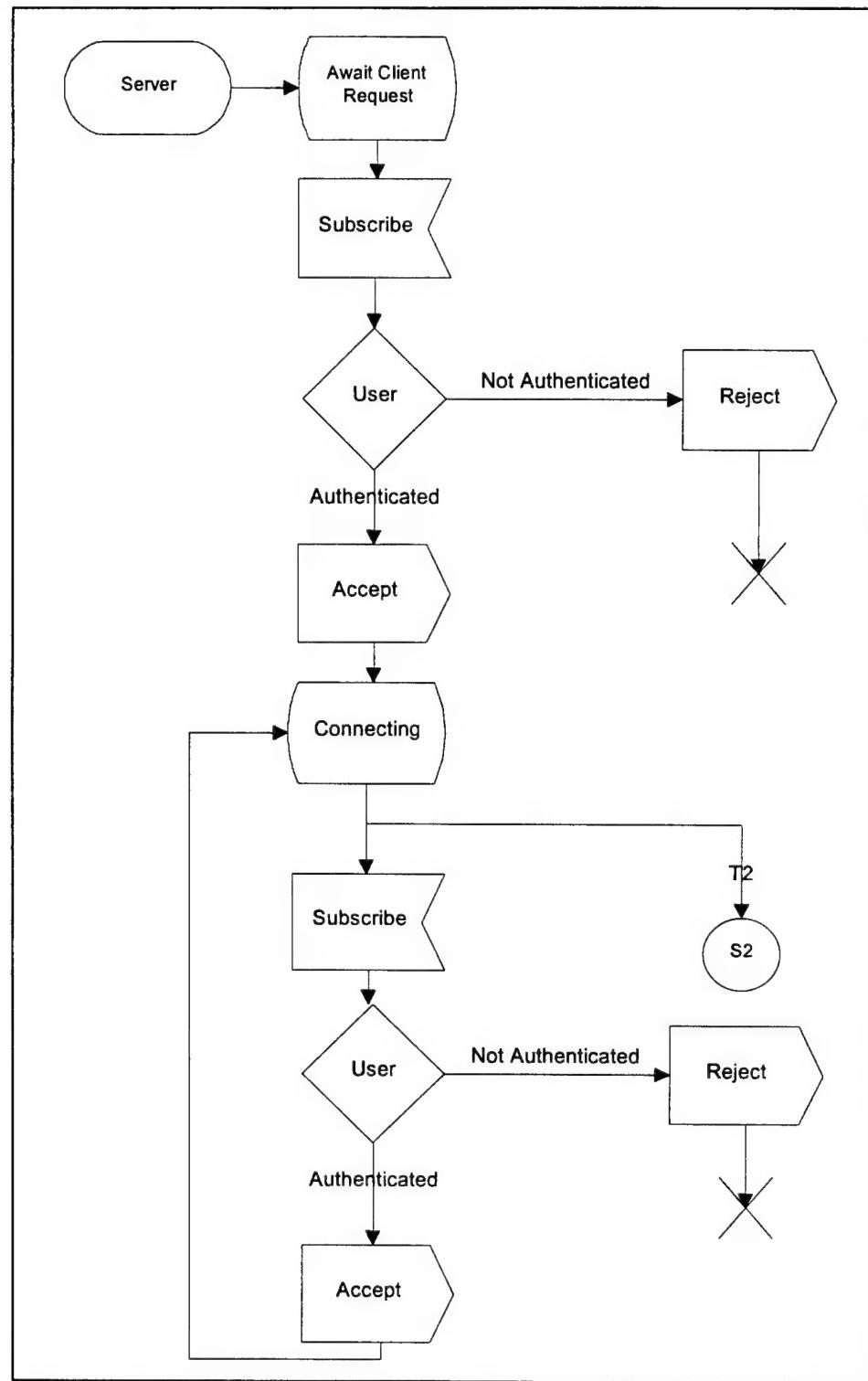
Figure 7-4. Graphic Representation Server (1 of 3).
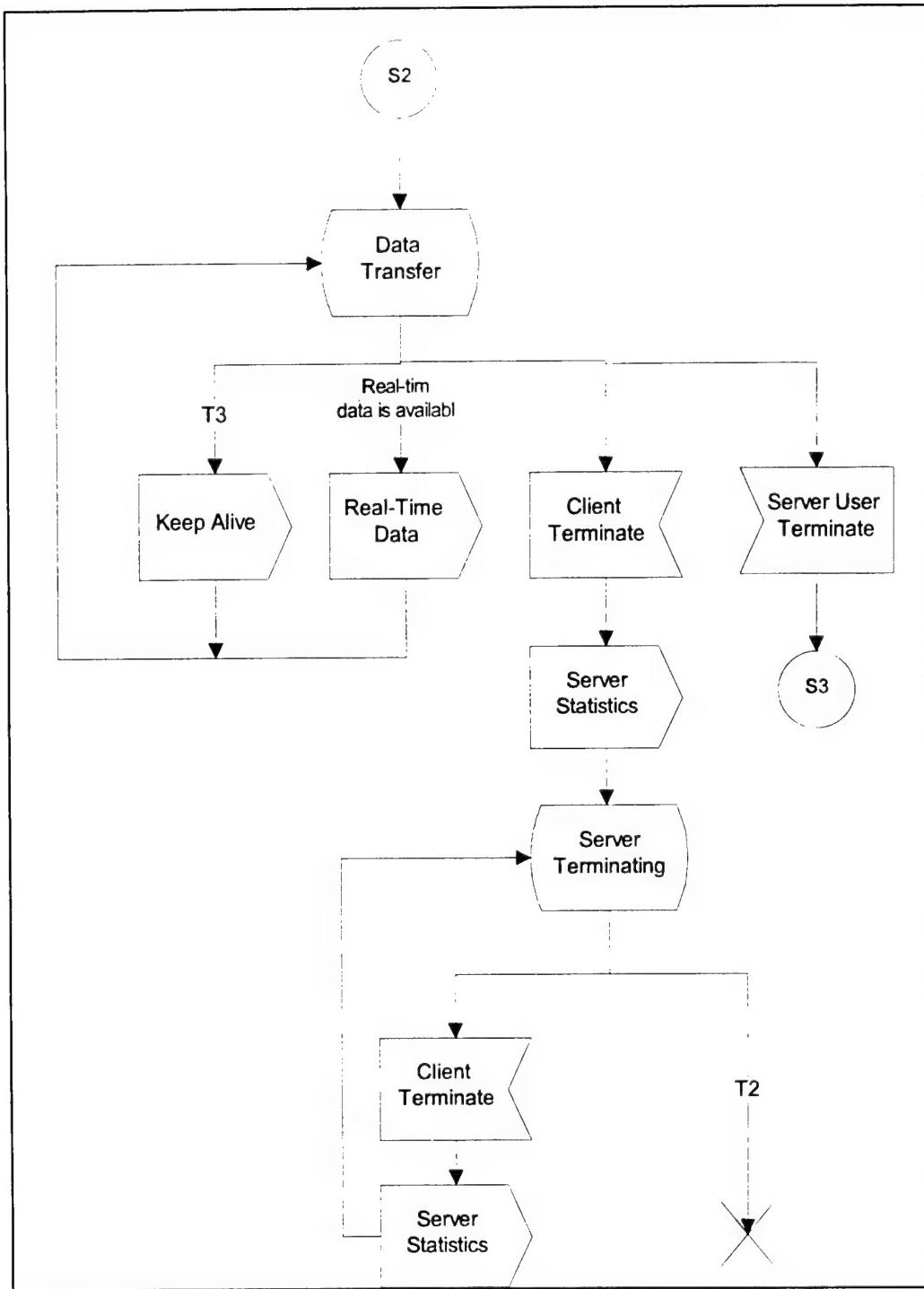
Figure 7-5. Graphic Representation Server (2 of 3).
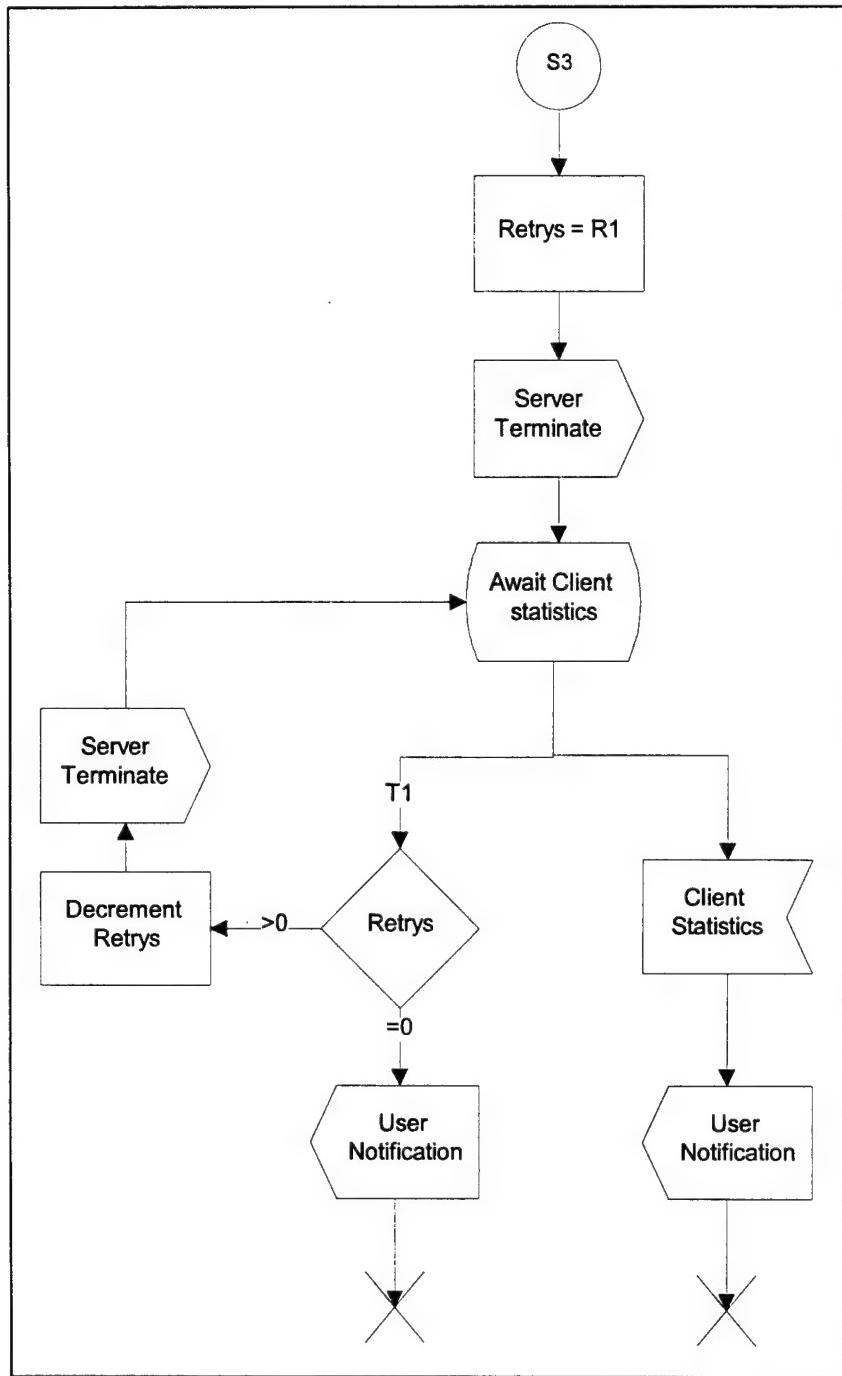
Figure 7-6. Graphic Representation Server (3 of 3).

CHAPTER 8.0

Protocol Data Units

Application Protocol Data Units (PDUs) are the data elements that are passed between the Client application and the Server application (both directions). These application layer PDUs are passed as a unit to the lower layers of communication software where they may be multiplexed with others, or broken into smaller segments, or other processing which is transparent to the Client and the Server. The important concept is that the PDU handed to the communication software by the Client or Server on one end is handed to the Server or Client on the other end as a unit. Thus we need not worry about delimiting the PDU with such things as "Start of Message" or "Synchronization Bits", these being handled by the lower layers.

The PDU is divided into 2 sections:

1. A *Header* section formatted identically for all PDUs.
2. A *Payload* section that contains the information associated with the PDU and whose formatting varies according to the PDU type.

The Syntax of the PDU with details of the header is shown in Table 8-1.

Table 8-1. PDU General Format

SECTION	FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header	PDU Type	1	Enum[4]	NA[5]	PDU Type. Types are defined in Table 8-2 and the enumeration coding is defined in Table 10-1.
Header	Size	2	Unsigned Integer	Octets	Length of PDU in Octets including header[6]
Header	Sequence Number	1	Unsigned Integer	NA	Monotonic and increasing by 1 for each PDU sent
Header	Classification	1	Enum	NA	Security Classification. Classification enumeration coding is defined in Table 10-5
Header	Session ID	1	Unsigned Integer	NA	Identifies the session, assigned by Server
Header	Spare	2		NA	For future expansion
Header	Time	4	R-Time[7]	millisec	Time this PDU header was generated, relative to the time word sent in the Accept PDU.
Payload					Payload for each PDU type is described in the subsequent sections

---

[4] Enumeration. Takes on one value in a list of values

[5] Not Any. This field is unitless.

[6] The length of the PDU is constrained by the UDP datagram size, which includes the 8-octet datagram header. See Reference 1 for details. This means the length of the IRIG STD 168-98 PDU including header can be no larger than 65535-8 or 65527 octets.

[7] See Table 9-1. R-Time Format

8.1 PDU Summary

Table 8-2. PDU Summary

PDU	ENUM VALUE	DESCRIPTION	FROM	TO
Accept	1	Positive response to Subscribe PDU	Server	Client
Client Statistics	2	The Client sends statistics relating to a session to the Server	Client	Server
Client Terminate	3	The Client notifies the Server that the Client wishes to terminate the session	Client	Server
Keep Alive	4	Sent by the Server when real-time data is not yet available. This prevents the Client from timing out and declaring an error.	Server	Client
Real-Time Data	5	PDU which contains real-time data	Server	Client
Reject	6	Notifies the Client that the Server could not honor the sub-scription request	Server	Client
Server Statistics	7	The Server sends statistics relating to a session to the Cli-ent	Server	Client
Server Terminate	8	The Server notifies the Client that the session is terminated and sends server statistical information	Server	Client
Subscribe	9	Subscribe to a particular mission, a particular data type, in a particular format.	Client	Server

8.2 Accept PDU

Table 8-3. Accept PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Time Reference	8	A-Time	See Table 9-2	Benchmark time for the time word used in the header. This time provides an absolute reference for the header time word, which is a *relative* time.
Time Source	1	Enum	NA	Specifies the time source used by the server in all time words in the header. This time source may or may not be the time source used in the real-time payload since the payload could have originated at another machine. (See Table 10-7 for enumeration)
Spare	1		NA	For future use
Static Data	Variable	Vari-able	NA	This section contains static data in PVL[8] that is passed to set up the session. The type of static data is de-pendent upon the Data Type and Format being trans-ferred. The following paragraphs contain details for this section

---

[8] PVL, Parameter Value Language. See Reference 4 and Reference 5.

8.2.1 SPI Accept PDU. There are a number of possible TSPI formats; however, the static data in the Accept PDU will be the same for all TSPI formats. This is defined below:

Table 8-4 TSPI Accept PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Time Reference	8	A-Time	See Table 9-2	Benchmark time for the time word used in the header. This time provides an absolute reference for the header time word, which is a *relative* time.
Time Source	1	Enum	NA	Specifies the time source used by the server in all time words in the header. This time source may or may not be the time source used in the real-time payload since the payload could have originated at another machine. (See Table 10-7 for the enumeration)
Spare	1		NA	For future use
TSPI Time Reference	8	A-Time	See Table 9-2	Benchmark time for the time word used in the real-time TSPI data. This time provides an absolute reference for the real-time time word, which is a *relative* time.
Static Parameters	Variable	Variable	NA	This area contains a series of parameter definitions in Parameter Value Language (See Reference 4 and Reference 5). As a minimum the following data elements (see Table 13-1) shall be defined: UserID MissionID RTOrgin RTOrientation In addition to these required data elements, the user may include additional parameter definitions as appropriate for the particular test.

The TSPI Accept PDU defines the coordinate system to be used in the passing of real-time data. This coordinate system is defined as the *real-time coordinate system* and it is defined by the Accept PDU in terms of the *world coordinate system*. See 0 for descriptions.

Specifying its location (RTOrigin) and its orientation (RTOrientation) relative to the world coordinate system define the real-time coordinate system. See Table 13-1 for further definitions of these data elements.

This approach allows a variety of real-time coordinate systems to be used. It provides the ability to use any local tangent plane or any geocentric coordinate system equally as well.

8.2.2 RAJPO Accept Data

8.3 Client Statistics PDU

Table 8-5. Client Statistics PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Total Rx	4	Unsigned Integer	PDUs	Total number of PDUs received
Sequence Rx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU received
Time Rx First	8	A-Time	See Table 9-2	The time the first PDU was received
Sequence Rx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU received
Time Rx Last	8	A-Time	See Table 9-2	The time the last PDU was received
Total Tx	4	Unsigned Integer	PDUs	Total number of PDUs transmitted (sent)
Sequence Tx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU transmitted (sent)
Time Tx First	8	A-Time	See Table 9-2	The time the first PDU was transmitted (sent)
Sequence Tx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU transmitted (sent)
Time Tx Last	8	A-Time	See Table 9-2	The time the last PDU was transmitted (sent)

8.4 Client Terminate PDU

Table 8-6. Client Terminate PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Term Reason	2	Enum	Unitless	Reason for Termination. See Table10-2
Total Rx	4	Unsigned Integer	PDUs	Total number of PDUs received
Sequence Rx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU received
Time Rx First	8	A-Time	See Table 9-2	The time the first PDU was received
Sequence Rx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU received
Time Rx Last	8	A-Time	See Table 9-2	The time the last PDU was received
Total Tx	4	Unsigned Integer	PDUs	Total number of PDUs transmitted (sent)
Sequence Tx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU transmitted (sent)
Time Tx First	8	A-Time	See Table 9-2	The time the first PDU was transmitted (sent)
Sequence Tx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU transmitted (sent)
Time Tx Last	8	A-Time	See Table 9-2	The time the last PDU was transmitted (sent)

8.5 Keep-Alive PDU

Table 8-7. Keep-Alive PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Time	8	A-Time	See Table 9-2	Time this PDU Payload was generated
NA Reason	2	Enum	See Table 10-4	Reason the Real-Time data is not available.

8.6 Real-Time Data PDU

Table 8-8. Real-Time Data PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Real-time Payload	variable			Payloads are defined by their associated data type and data formats. These are called out in the "Accept" PDU and are static for a given session. The session ID in the header of each "Real-Time Data" PDU is used by the Client to identify the session and thus to determine the data type and format of the payload. See section 10-2.

8.7 Reject PDU

Table 8-9. Reject PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Reject Reason	2	Enum	Unitless	SeeTable 10-3.
Data Type	2	Enum	NA	Specifies the data type requested. See Table 10-6 for a list of data types
Data Format	2	Enum	NA	Specifies the requested data format. See section 10.2 for a list of data formats.
Time	8	A-Time	See Table 9-2	Time this PDU Payload was generated
Static Parameters	Variable	Variable	NA	This area contains a series of parameter definitions in Parameter Value Language (See Reference 4 and Reference 5). As a minimum the following data elements (see Table 13-1) shall be defined: UserID MissionID In addition to these required data elements, the user may include additional parameter definitions as appropriate for the particular test.

8.8 Server Statistics PDU

Table 8-10. Server Statistics PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Total Rx	4	Unsigned Integer	PDUs	Total number of PDUs received
Sequence Rx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU received
Time Rx First	8	A-Time	See Table 9-2	The time the first PDU was received
Sequence Rx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU received
Time Rx Last	8	A-Time	See Table 9-2	The time the last PDU was received
Total Tx	4	Unsigned Integer	PDUs	Total number of PDUs transmitted (sent)
Sequence Tx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU transmitted (sent)
Time Tx First	8	A-Time	See Table 9-2	The time the first PDU was transmitted (sent)
Sequence Tx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU transmitted (sent)
Time Tx Last	8	A-Time	See Table 9-2	The time the last PDU was transmitted (sent)

8.9 Server Terminate PDU

Table 8-11 Server Terminate PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Term Reason	2	Enum	Unitless	Reason for Termination. See Table 10-2
Total Rx	4	Unsigned Integer	PDUs	Total number of PDUs received
Sequence Rx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU received
Time Rx First	8	A-Time	See Table 9-2	The time the first PDU was received
Sequence Rx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU received
Time Rx Last	8	A-Time	See Table 9-2	The time the last PDU was received
Total Tx	4	Unsigned Integer	PDUs	Total number of PDUs transmitted (sent)
Sequence Tx First	2	Unsigned Integer	Unitless	The sequence number of the first PDU transmitted (sent)
Time Tx First	8	A-Time	See Table 9-2	The time the first PDU was transmitted (sent)
Sequence Tx Last	2	Unsigned Integer	Unitless	The sequence number of the last PDU transmitted (sent)
Time Tx Last	8	A-Time	See Table 9-2	The time the last PDU was transmitted (sent)

8.10 Subscribe PDU

Table 8-12. Subscribe PDU

FIELD	OCTETS	TYPE	UNITS	DESCRIPTION
Header				See Table 8-1
Data Type	2	Enum	NA	Specifies the data type requested. See Table 10-6 for a list of data types
Data Format	2	Enum	NA	Specifies the requested data format. See section 10.2 for a list of data formats.
Time	1	Enum	NA	Specifies the time source used by the client in all time words in the header.
Spare	3		NA	For future use
Static Parameters	Variable	Variable	NA	This area contains a series of parameter definitions in Parameter Value Language (See Reference 4 and Reference 5). As a minimum the following data elements (see Table 13-1) shall be defined: UserID Authentication MissionID In addition to these required data elements, the user may include additional parameter definitions as appropriate for the particular test.

Note: The Session ID is not known when the Subscribe PDU is generated, consequently, this field (in the PDU header) cannot be determined. For the Subscribe PDU, the Session ID field is set to zero.

CHAPTER 9.0

Format Notes

General Comments on Time Stamp

It is important to note that the time in the header is not intended to be the time of a *payload sample*, though the two might be related or even identical. The time stamp in the IRIG STD 168-98 PDU header is intended to be used for services such as archiving, sorting, processing and correlation with other data sets. Time stamp of a *payload* is best understood by examining TSPI (Time, Space, Position Information) data. Here it is important to realize that the TSPI sample has 4 independent *measurements* (time being one of them). One can, and often does, adjust the time (or epoch) of a TSPI sample to accommodate such considerations as latency or differences in sampling and processing rates. Thus, the time word in these samples may be subject to processing which is not a part of, or not known by, the IRIG 168-98 software. The time word in the PDU header is the time the PDU was generated by the IRIG 168-98 software and since the data is real-time data it is expected that this will be very close to, but not necessarily equal to, the time associated with any data samples.

There are two time formats used in the IRIG STD 168-98 protocol:

1. R-Time - Relative Time used in the PDU header and thus appears in all PDUs
2. A-Time -Absolute Time used in some IRIG STD 168-98 control and setup PDUs

9.1 R-Time Format

To accommodate the need to minimize the time stamp in the header and yet provide good resolution and dynamic range, a 32 bit time stamp format is chosen. The units are binary milliseconds (value of the least significant bit) and the dynamic range then becomes 1193 hours (almost 50 days). This is shown in Table 9-1 below. The R-Time is a *Relative* time from a benchmark. This benchmark time is established by a variety of means depending on the use of the R-Time Format. For example, the R-Time Formatted time word in the PDU header is relative to a benchmark time established in the Accept PDU.

Table 9-1. R-Time Format

Equal to 2,147,483.648 seconds

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSB milliseconds															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LSB milliseconds															

Equal to 1 millisecond

9-1

9.2 A-Time Format

A-Time is used to specify a time with microsecond resolution, but also includes enough information to specify the year/day/hour as well. A-Time is used in some IRIG STD 168-98 control PDUs to fully specify the time. Table 9-2 shows the format and the discussions with each PDU (see the PDU definitions in CHAPTER 8.0) explain the meaning.

Table 9-2. A-Time Format

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		X							Year						
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X					Day of Year							Hour		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			Minutes					Seconds					MSB microseconds		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
							LSB microseconds								

CHAPTER 10.0

Enumeration Tables

10.1 Message Type

Table10-1. Message Type Enumeration

VALUE	MEANING
1	Accept PDU
2	Client Statistics PDU
3	Client Terminate PDU
4	Keep-Alive PDU
5	Real-Time Data PDU
6	Reject PDU
7	Server Statistics PDU
8	Server Terminate PDU
9	Subscribe PDU

10.2 Terminate Reason

This field defines the reason the session is being terminated.

Table 10-2. Terminate Reason Enumeration Table

VALUE	MEANING
0	Unknown
1	User on the Client side requested termination
2	User on the Server side requested termination
3	Mission complete

10.3 Reject Reason

This field defines the reason the subscribe request is being rejected.

Table 10-3. Reject Reason Enumeration Table

VALUE	MEANING
0	Unknown
1	User ID unknown
2	User not authorized
3	Mission not available
4	Data type for specified mission not available
5	Data format for specific data type not available

10.4 Data Not Available Reason

This field defines the reason data is not available and thus the need to send a "Keep-Alive" PDU.

Table 10-4. Data Not Available Enumeration Table

VALUE	MEANING
0	Unknown
1	The real-time data didn't arrive in time to send before the expiration of the timer
2	The real-time data source is not yet available (mission not yet started)
3	The real-time data is temporarily not available
4	The real-time data source has experienced a malfunction and thus data is not available at this time.

10.5 Security Classification Enumeration

Table 10-5. Security Classification Enumeration Table

VALUE	MEANING
1	UNCLASSIFIED
2	CONFIDENTIAL
3	SECRET
4	TOP SECRET

10.6 Data Type Enumeration

Table 10-6. Data Type Enumeration Table

VALUE	MEANING	FORMAT ENUMERATION IN
1	Test Pattern	
2	Time Space Position Information (TSPI)	See Table 10-9
3	RAJPO	
4	Telemetry	
5	Digitized Voice	

10.7 Time Source Enumeration

Table 10-7. Time Source Enumeration Table

VALUE	MEANING	FORMAT ENUMERATION IN
1	Time is Synchronized to Universal Time Co-ordinated (UTC)	Time Formats are specified in the various PDU definitions
2	Time is Synchronized to a local clock which may or may not be synchronized to UTC	Time Formats are specified in the various PDU definitions
3	Time is based on the Computer Real-Time Clock	Time Formats are specified in the various PDU definitions

10.8 Formats for Each Data Type

10.8.1 Test Pattern Formats

Table 10-8. Test Pattern Format Enumeration Table

VALUE	MEANING	FORMATS DEFINED IN
1	Quick Brown Fox	Section 11.1.1 on page 11-1

10.8.2 TSPI Formats

10.8.2.1 TSPI Format Enumeration

Table 10-9. TSPI Formats Enumeration Table

VALUE	MEANING	FORMATS DEFINED IN
1	Universal TSPI 3 DOF[9] Low Resolution	Section 11.2.1
2	Universal TSPI 3 DOF High Resolution	Section 11.2.2
3	Universal TSPI 6 DOF Low Resolution	Section 11.2.3
4	Universal TSPI 6 DOF High Resolution	Section 11.2.4

10.8.2.2 TSPI Mode Enumeration

Table 10-10. TSPI Mode Enumeration Table

VALUE	MEANING
1	TSPI Data Available
2	TSPI Data from valid real-time track
3	TSPI Data from computer generated theoretical trajectory
4	TSPI Data from computer generated points based on historical track data.

10.8.3 RAJPO Formats

Table 10-11. RAJPO Formats Enumeration Table

VALUE	MEANING	FORMATS DEFINED IN
1	Universal RAJPO	TBD

10.8.4 Telemetry Formats
TBD

10.8.5 Digitized Voice Formats
TBD

---

[9] DOF - Degrees of Freedom

CHAPTER 11.0

Payload Formats for Each Data Type

11.1 Test Pattern

11.1.1 Quick Brown Fox

The payload for this test pattern is an ASCI character string that includes the following: "The quick brown fox jumped over the lazy dog's back."

11.2 Universal TSPI Formats

11.2.1 Universal TSPI 3 DOF low Resolution

Table 11-1. Universal TSPI 3 DOF Low Resolution

SYMBOL	PARAMETER	OCTETS	FORMAT	UNITS
T	Time Stamp	4	R-Time	See Table 9-1
X	1st Cartesian position Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
Y	2nd Cartesian position Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
Z	3rd Cartesian position Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{X}$	1st Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{Y}$	2nd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{Z}$	3rd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
S	Data Source	1	Enum	See Table 10-10
Q	Data Quality	1	Binary	0 to 255, where 0 is worst and 255 is best.
	Spare	2	N/A	N/A

11.2.2 Universal TSPI 3 DOF High Resolution

Table 11-2. Universal TSPI 3 DOF High Resolution

SYMBOL	PARAMETER	OCTETS	FORMAT	UNITS
T	Time Stamp	4	R-Time	See Table 9-1
X	1st Cartesian position Component	8	IEEE Floating Point Double Precision. See Reference 3.	Meters
Y	2nd Cartesian position Component	8	IEEE Floating Point Double Precision. See Reference 3.	Meters
Z	3rd Cartesian position Component	8	IEEE Floating Point Double Precision. See Reference 3.	Meters
$\dot{X}$	1st Cartesian velocity Component	4	IEEE Floating Point Singel Precision. See Reference 3.	Meters
$\dot{Y}$	2nd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{Z}$	3rd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
S	Data Source	1	Enum	See Table 10-10
Q	Data Quality	1	Binary	0 to 255, where 0 is worst and 255 is best.
	Spare	2	N/A	N/A

11.2.3 Universal TSPI 6 DOF Low Resolution

Table 11-3. Universal TSPI 6 DOF Low Resolution

SYMBOL	PARAMETER	OCTETS	FORMAT	UNITS
T	Time Stamp	4	R-Time	See Table 9-1
X	1st Cartesian position Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
Y	2nd Cartesian position Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
Z	3rd Cartesian position Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{X}$	1st Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{Y}$	2nd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{Z}$	3rd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
S	Data Source	1	Enum	See Table 10-10
Q	Data Quality	1	Binary	0 to 255, where 0 is worst and 255 is best.
	Spare	2	N/A	N/A
			<TBD> add orientation fields	

11.2.4 Universal TSPI 6 DOF High Resolution

Table 11-4. Universal TSPI 6 DOF High Resolution

SYMBOL	PARAMETER	OCTETS	FORMAT	UNITS
T	Time Stamp	4	R-Time	See Table 9-1
X	1st Cartesian position Component	8	IEEE Floating Point Double Precision. See Reference 3.	Meters
Y	2nd Cartesian position Component	8	IEEE Floating Point Double Precision. See Reference 3.	Meters
Z	3rd Cartesian position Component	8	IEEE Floating Point Double Precision. See Reference 3.	Meters
$\dot{X}$	1st Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{Y}$	2nd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
$\dot{Z}$	3rd Cartesian velocity Component	4	IEEE Floating Point Single Precision. See Reference 3.	Meters
S	Data Source	1	Enum	See Table 10-10
Q	Data Quality	1	Binary	0 to 255, where 0 is worst and 255 is best.
	Spare	2	N/A	N/A
			<TBD> add orientation fields	

11.3 Universal RAJPO Format

TBD

11.4 Telemetry Formats

TBD

11.5 Digitized Voice Formats

TBD

CHAPTER 12.0

Protocol Parameters

12.1 Protocol Timers

Table 12-1. Protocol Timers

TIMER	DESCRIPTION	VALUE
T1	Time to wait for response to a PDU requiring a response.	1 sec
T2	Time to wait after sending a response PDU to make sure the response was received.	(R1+1)*T1
T3	Maximum time the Server waits between sending "Real-Time Data" PDUs. After this time a "Keep-alive" PDU is sent	5 sec
T4	Maximum time the Client expects between Real-Time Data PDUs (or "Keep-Alive" PDUs). After this time expires without receiving a PDU from the Server, the Client notes an error. If the errors exceed a maximum threshold (implementation dependent) the Client notifies the User. The User is then expected to terminate the session.	1.5 * T3

12.2 Protocol Retries

Table 12-2. Protocol Retries

RETRY	DESCRIPTION	VALUE
R1	When sending a PDU that requires a response, a timer is set (T1). Upon expiration of this timer, the PDU is retransmitted and the timer is set again. R1 sets the maximum number of times this will be repeated.	4

CHAPTER 13.0

Data Element Dictionary

A number of data elements are defined in the following Data Element Dictionary (DED). These data elements are required as part of the protocol. They are required to be specified in a number of PDUs using the Parameter Value Language (PVL) as described in Reference 4 and Reference 5.

Table 13-1. Data Element Dictionary

Data Element	Type	Description	Units
UserID	Text	User Identification. This field along with the Authentication information is used to determine if the requester is authorized to receive the requested data.	N/A
Authentication	Text	Authentication Text. This information is used by the Server to validate the Client. Each server will establish authentication procedures. The default for this field is the encrypted value of a user password. The encryption algorithm is TBD.	N/A
MissionID	Text	The mission ID for which data is requested.	N/A
RTOrigin	Sequence	Defines the origin of the real-time coordinate system to be used in the subsequent transmission of real-time position and velocity data. Consists of a sequence of three numbers representing the three vector components of a location vector defining the origin of the real-time coordinate system. The location vector is defined in world coordinates (earth fixed geocentric). See 8.2.1 SPI Accept PDU on page 8-3 for more descriptions.	Meters
RTOrientation	Sequence	Defines the orientation of the real-time coordinate system to be used in the subsequent transmission of real-time position and velocity data. Consists of a sequence of three numbers representing the three Euler angles specifying the orientation of the real-time coordinate system relative to the world coordinate system. See 8.2.1 SPI Accept PDU on page 8-3 for more descriptions.	Radians

CHAPTER 14.0

Coordinate Systems (Used in Chapter 13)

The TSPI formats used herein are based on the concept of a *world coordinate system* as shown in Figure 14-1. This coordinate system is in common use throughout the member RCC ranges. It is often referred to as the EFG. EFG stands for earth fixed geocentric and also represents the symbols most commonly used for its principal axes. It's origin is at the geocenter. The positive E axis extends from the origin through the prime meridian at the equator. The positive G axis extends from the origin through the north axis. The F axis makes it a right-handed Cartesian coordinate system (i.e. $\hat{E} \times \hat{F} = \hat{G}$).
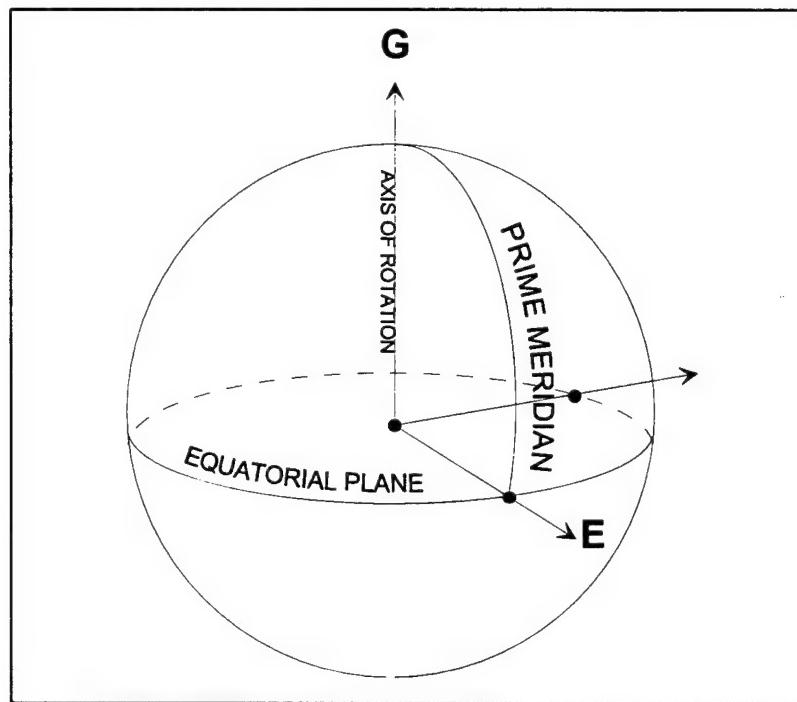
Figure 14-1. World Coordinates (EFG).

The TSPI formats defined herein use a *real-time coordinate system*, which is defined in terms of the world coordinate system during the session initiation. This real-time coordinate system is defined by specifying the *location* of it's origin and its *orientation* relative to the world coordinate system. This permits a great deal of flexibility in the choice of the real-time coordinate system. The approach allows one to use either geocentric coordinates (by setting the location to (0,0,0)) or to use topocentric coordinates by choosing some other location (such as a reference point on the range). In some situations, such as orbital tracking, geocentric coordinates are preferred and in others, such as local range activity, topocentric coordinates work better. The approach described herein works equally well in either case.

To define orientation, Euler angles, ϕ, θ, and ψ, are used. This is a method of defining the orientation of one coordinate system to another by specifying a series of three rotations and is well documented in the literature. However the designation of the Euler angles and even the manner in which they are generated, are not universally agreed upon. The notation used here is that most commonly found in modern texts. Briefly, this is described below.

Starting with the two coordinate systems (world coordinates, EFG, and real-time coordinates, XYZ) aligned:

1. First, perform a counterclockwise rotation through an angle ϕ about the Z axis.
2. Then, perform a counterclockwise rotation through an angle θ about the new X axis.
3. Finally, perform a counterclockwise rotation through an angle ψ about the new Z axis.

REFERENCES

Reference 1   Internet Protocol Suite, RFC 793, Postel, J. Transmission Control Protocol. 1981 September; 85 p.

Reference 2   Internet Protocol Suite, RFC 768 Postel, J. User Datagram Protocol. 1980 August 28; 3 p.

Reference 3   IEEE Computer Society (1985), "IEEE Standard for Binary Floating-Point Arithmetic", IEEE Std 754-1985.

Reference 4   "Recommendation for Space Data Systems: Parameter Value Language Specification", CCSDS 641.0-B-1, Consultative Committee for Space Data Systems, Blue Book, Issue 1, May 1992.

Reference 5   "Report Concerning Space Data System Standards: Parameter Value Language -- A Tutorial", CCSDS 641.0-G-1, Consultative Committee for Space Data Systems, Green Book, Issue 1, May 1992.

Reference 6   "Specification and Description Language" ITU Z.100 SDL-92, ITU, Geneva, 1994.

APPENDIX A

MESSAGE SEQUENCE CHARTS

MESSAGE SEQUENCE CHARTS

Message Sequence Charts (MSCs) are an International Telecommunications Union (ITU) recommendation (Z.120) and a widespread means for the visualization of selected system runs (traces) within communication systems. MSCs focus on the communication behavior of system components and their environment by means of message exchange. MSCs provide a clear description of system traces in the form of message flow diagrams. MSCs include both a textual (MSC/PR) and graphical representation (MSC/GR) syntactical form. For purposes of this specification we will only use the MSC/GR form.

The basic language of MSCs includes all constructs that are necessary in order to specify the pure message flow. For MSCs these language constructs are instance, message, environment, action, timer set, timer reset, time-out, instance creation, instance stop, and condition. Each of these basic constructs is described below.

Instance - Represented by vertical lines (or optionally vertical columns).   or
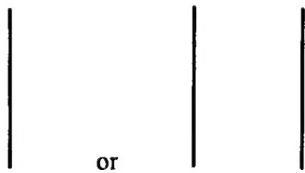
Message- Represented by horizontal arrows (or with a downward slope or bent to admit message overtaking or crossing).

or   or

Corrupted Message - Represented by a dashed arrow.
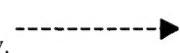
Environment - Represented by a frame that forms the boundary of the MSC diagram.

Protocol Entity - Represented by a rectangle containing name of protocol entity.   Client
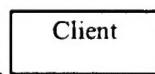
Timer - Represented by the symbol   with the bottom bent arrow being the Time Out representation.
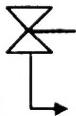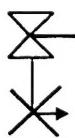
Timer Reset - Represented by the symbol

Terminate Instance - Represented by a solid block at the bottom of the instance.

Summary Description - Representing a series of interactions at a summary level
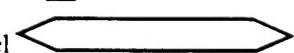
APPENDIX B

SPECIFICATION AND DESCRIPTION
LANGUAGE/GRAPHIC REPRESENTATION (SDL/GR)

Specification and Description Language/Graphic Representation (SDL/GR)

IRIG 168-98 defines the behavior of the Client Protocol Entity and the Server Protocol Entity using a graphic representation. This graphic representation is based on the CCITT[10] Specification and Description Language (SDL-92). SDL-92 is a very extensive language for specifying systems and the behavior of systems and includes features far beyond what is needed for the protocol described in IRIG 168-98. Therefore only a small subset of SDL-92 is used in the behavior specifications and that can be described easily herein. The relevant symbols are shown in Table B-1. The graphic representation is a combination of traditional logic flow charts and state transition diagrams. The logic flow description will be familiar to most computer professionals. These charts have an added feature, not often seen in traditional logic flowcharts, they also depict a state machine showing the states and transitions between states for the protocol entities.

The *state* symbol, ⌐⎯⌐, is used to indicate a pause in the logic where the logic waits until one of several events takes place. One or more alternative logic paths exiting from the state symbol indicate the possible events that can cause a state transition. In the example shown in Figure B-1 below, the "Connecting" state is entered and the protocol entity remains in this state until timer "T1" expires, a "Reject" PDU is received, or an "Accept" PDU is received.
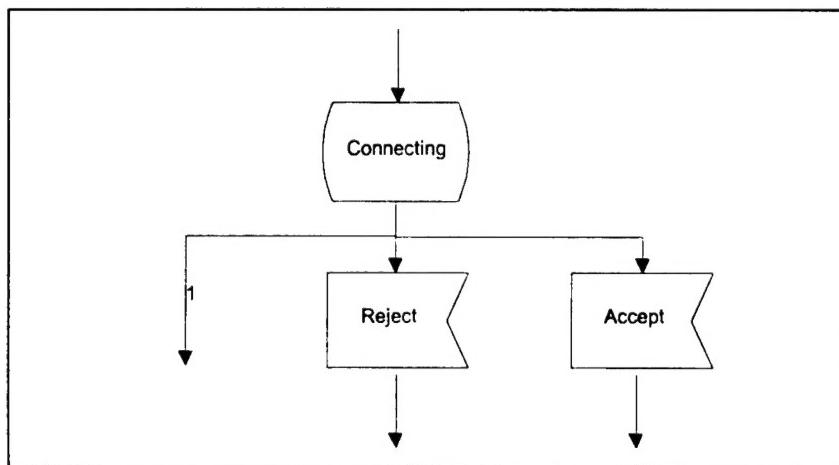

Figure B-1 Graphical Respresentation Fragment

---

[10] The Name CCITT was replaced in 1993 by the name ITU-T (International Telecommunications Union – Telecommunications Standardization sector).

Table B-1. Graphical Symbols
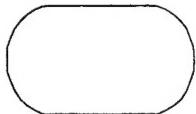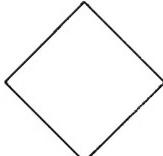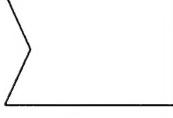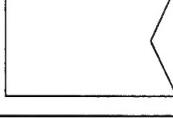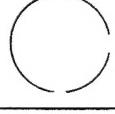
Symbol	Description
	Start The starting point or entry point.
	Process Indicates processing performed as defined by the text included within the symbol.
	State Idicates a state in which the Protocol Entity remains until an event occurs. The possible events which can occur are indicated by the logic paths out of this symbol
	Decision Indicates one of several alternative paths for the logic flow. The possible paths are indicated by the logic paths out of this symbol
	Output to Network Side Indicates an output to the network side. Typically, this will be a PDU to the peer level protocol entity in the remote machine.
	Output to User Side Outputs to the user are typically notifications of events that have occurred. These may be expected or unexpected events.
	Input from User Side The user-side input typically consists of an event representing an operator action such as "start a session", or "terminate a session".
	Input from Network Side Idicates the arrival of an input from the network side. Typically, this will be a PDU received from the peer level protocol entity on the remote machine.
	Connector Typically used to connect logic paths across pages.
	Terminate Used to indicate the end of the logic path.